Escuela Universitaria Ingeniería Técnica Industrial ZARAGOZA

PROYECTO FIN DE CARRERA

CONTROL CON MICROCONTROLADOR DE UN PROTOTIPO DE VOLANTE DE INERCIA LEVITADO POR SUPERCONDUCTORES

Autor: Alberto Aparicio Miñana
Directores: Jesús Letosa Fleta
Mario Mora Alfonso
Zaragoza. Septiembre 2010.

“A mi familia, a mis
padres y a mi hermana.

A mis directores de proyecto:
D. Jesús Letosa
D. Mario Mora
Los cuales me han permitido
embarcarme en este proyecto
 juntos hemos conseguido los
objetivos propuestos no sin sufrir
quebraderos de cabeza

A D.Joaquín Mur, maestro de taller
del Dep de ingeniería eléctrica,
que me facilito los ensayos
mediante material y nos suministro
el Spray aislador.

Al personal de laboratorio del
departamento de ingeniería
eléctrica:
D. Juan A. Tejero Goméz
D. Alfredo Benedicto Exposito
D. Carlos Fuertes Torre
a los que a lo largo del proyecto
he estado interrumpiendo en sus
quehaceres pidiéndoles
material y ayuda, siendo siempre
muy bien atendido.

A los maestros del taller de
materiales:
D. Israel Cabistany García
D. José A. Gómez García
que me ayudaron a realizar
 las modificaciones en el prototipo
los cuales siempre estuvieron
dispuestos a echarme una mano y
siguieron muy de cerca la
evolución del prototipo

A todos les doy las gracias

Índice

1. Introducción

Este proyecto, es la continuación de otros proyecto fin de carrera anteriores que han añadido elementos al proyecto final de carrera desarrollado por D. Pedro J. Lambea "Prototipo de volante de inercia levitado mediante superconductores", el proyecto en el que nos encontramos ahora trata de su control mediante un microcontrolador.

El objetivo final de este prototipo es convertirlo en un sistema de almacenamiento y recuperación de energía. Actualmente solo está en funcionamiento la parte de almacenamiento, se tiene la intención que próximos proyectos logren recuperar la energía cinética del rotor convirtiéndola en eléctrica.

Lo que se ha conseguido con la realización de este proyecto es que el prototipo aumente de velocidad hasta su velocidad máxima de forma automática, a la cual se desconecta de la red. Entonces el volante sigue girando por su propia inercia, por el rozamiento con el aire y con los cristales de hielo. El rotor se frena hasta alcanzar la velocidad mínima a la cual se conecta a red para que alcance de nuevo la velocidad máxima, y lo mantenemos dentro de este rango de velocidad.

Cuando este realizada la parte de recuperación, al conectar una carga el prototipo cederá la energía cinética acumulada para generar energía eléctrica lo cual hará que se frene. Al alcanzar la velocidad mínima volvería a conectarse a la red para volver a cargarse.

Se ha empleado un microcontrolador para que se encargue de las funciones que son necesarias para controlar el prototipo.

Además se ha diseñado y construido un sistema de anclaje del rotor cuando los superconductores no estén en activos, este estado ocurre cuando no se usa el prototipo, y cuando están activos lo sustenta a la distancia optima para que el rotor pueda levitar y que evita que el estator al ser alimentado lo atraiga. Es el microcontrolador el cual controla este sistema para su funcionamiento de forma automática

A lo largo de esta memoria se exponen los diseños electrónicos y mecánicos realizados y la programación del microcontrolador.

2. OBJETIVOS

El objetivo principal es el control del rotor manteniéndolo en un rango de velocidad. Para conseguir esto empleamos un microcontrolador, el cual según las revoluciones a la que está girando el rotor, adopta las decisiones conexión y desconexión de la alimentación.

Diseño de un sensor de revoluciones que informe al microcontrolador de la velocidad que hay en cada momento en el rotor.

Colocación de un sistema de anclaje del rotor el cual se encarga de la función de sustentar el rotor. Tanto cuando el prototipo esta en uso que evita que el rotor sea atraído por el estator, esto se produce en el momento de la carga y para que lo coloque de nuevo en la posición correcta para que el rotor pueda levitar, como cuando no se utiliza el prototipo, al tenerlo almacenado o se está trasladando al lugar de la demostración, el sistema sujeta el rotor impidiendo que este pueda rozar o golpear en ningún lugar. Mediante este sistema sea conseguido reducir el rozamiento del rotor ya que hasta la fecha para evitar que fuera atraído se usaban unos centradores que ejercían fuerza sobre el eje del rotor para que este no rozara con el estator.

Diseño de la etapa de control del sistema de anclaje, este sistema interconecta el microcontrolador con el sistema de anclaje y dependiendo de la velocidad que tengamos se le mandara una orden determinada para que adopte una de las dos posiciones que tiene el sistema.

3. Microprocesadores

3.1 Introducción

Antes de nada voy a poner un poco los antecedentes de los microcontroladores para hacernos una idea de su evolución, repercusiones en la industria, explicar un poco su estructura de funcionamiento y lo que son capaces de aportar a nuestro proyecto

3.2 Historia de los microprocesadores

Hasta hace unos 40 años no existían los microprocesadores, se diseñaban circuitos electrónicos para cada aplicación constituidos por muchos componentes electrónicos y había que realiza cálculos matemáticos complejos. Estos circuitos lógicos utilizaban muchos elementos discretos (tampoco tenían los tamaños tan reducidos de los actuales, por lo que los tamaños de los circuitos eran considerables), tales como transistores, resistencias, diodos, condensadores, bobinas, etc..... Lo normal era que se produjeran muchos fallos a causa de tolerancias, ruidos en el sistema, fallos de los mismos componentes.... teniendo que ser ajustados introduciendo mas componentes los cuales complicaban aun más el diseño. Era impensable que pudiesen hacer tareas diferentes a aquellas para las que habían sido diseñadas, para ello había que cambiar la mayoría de sus elementos, más bien era construirlos de nuevo.

Una empresa japonesa BUSICOM le encargó a una incipiente compañía norteamericana que le produjera circuitos para una calculadora de bajo coste. INTEL desarrollo varios modelos de circuitos para ese propósito, entre los circuitos encargados había uno especial: el primer microprocesador integrado. BUSICOM no se mostró interesada en el invento, ya que entre las opciones había soluciones más fáciles y más baratas. INTEL sí supo ver el potencial del invento y les compró los derechos para comerciarlo.

En 1971 apareció en el mercado el 4004, una máquina digital sincrónica compleja. Un microprocesador de 4 bits que podía efectuar 6000 operaciones por segundo. TEXAS INSTRUMENTS fue la empresa que sintió interés por este producto. Ambas empresas trabajaron juntas para desarrollar en 1972 el 8008, con 16Kb de memoria, 45 instrucciones y capaz de efectuar 300.000 operaciones por segundo.

Otras empresas se sumaron a desarrollar microprocesadores propios para hacer competencia a INTEL tales como MOTOROLA, WESCON, ZILOG, LA TECNOLOGIA MOS.... así se inicio una revolución sin parangón en el avance de la tecnología. Introduciéndose rápidamente en todos los ámbitos, tras esto aun mas empresas se unieron a su desarrollo, ningún invento en la historia ha evolucionando a tal velocidad.

En la actualidad el avance de los microprocesadores a dado como resultado a los microcontroladores, capaces de dar inteligencia a cualquier maquina. Se los puede programar para que hagan cualquier tarea, responder a condiciones cambiantes y que responda a las necesidades propias de sus operarios.

3.3 ¿Qué es un microprocesador?

Un microprocesador depende de algunos circuitos integrados adicionales externos como por ejemplo: memorias RAM para almacenar los datos temporalmente y memorias ROM para almacenar el programa que se encargaría del proceso del equipo, un circuito integrado para los puertos de entrada y salida y un decodificador de direcciones.

3.4 ¿Qué es un microcontrolador?

Un microcontrolador es un circuito integrado programable que básicamente incluye 4 bloques: una CPU (unidad central de procesamiento) una memoria ROM y una memoria RAM, y unos puertos de entrada y salida (Inputs/Outputs). Este dispositivo electrónico es capaz de llevar a cabo procesos lógicos. Estos procesos o acciones son programados por el usuario a través de un programador, como ejemplo pongo el que estoy usando el Codewarrior aunque existen muchos mas según el microcontrolador que estemos usando, cada marca desarrolla sus propios programas para optimizar los recursos de sus productos, además existen distintos lenguajes de programación (Ensamblador, C, C++)

El microcontrolador es un circuito integrado que incluye todos los componentes de un computador pero no su tamaño y por esto es posible montar el controlador en el propio dispositivo al que gobierna. (embedded controller).

3.5 Los microcontroladores actuales

Los microcontroladores están a nuestro alrededor sin que seamos conscientes de ellos, unos realizan funciones sencillas y otros llevan a cabo procesos complejos. Los podemos encontrar controlando el funcionamiento de los ratones y teclados de los ordenadores, en los teléfonos, en los microondas, en los televisores.... El siglo XXI será testigo de la conquista de estos diminutos computadores, que gobernarán la mayor parte de los aparatos que fabricaremos y usaremos los humanos.

Cada vez existen más productos que incorporan un microcontrolador con el fin de aumentar sustancialmente sus prestaciones, reducir su tamaño y coste, mejorar su fiabilidad y disminuir el consumo.

Algunos fabricantes de microcontroladores superan el millón de unidades de un modelo determinado producidas en una semana. Este dato puede dar una idea de la masiva utilización de estos componentes.

Los microcontroladores están siendo empleados en multitud de sistemas presentes en nuestra vida diaria, como pueden ser juguetes, microondas, frigoríficos, televisores, computadoras, impresoras, módems, el sistema de arranque de nuestro coche, etc. Y otras aplicaciones con las que seguramente no estaremos tan familiarizados como instrumentación electrónica, control de sistemas en una nave espacial, etc. Una aplicación típica podría emplear varios microcontroladores para controlar pequeñas partes del sistema. Estos pequeños controladores podrían comunicarse entre ellos y con un procesador central, probablemente más potente, para compartir la información y coordinar sus acciones, como ocurre en cualquier PC.

3.6 El microcontrolador que vamos a usar

Ya visto que nos ofrece usar un microcontrolador ahora vamos a presentar el que vamos a utilizar. En el mercado existen muchísimos microcontroladores unos más sencillos y otros más complejos dependen mucho de para que van a ser usados, no merece la pena poner microcontroladores complejos que después tengan que realizar procesos sencillos ya que cuantas más prestaciones puedan ofrecer más se encarece su precio. Así que hay que elegir el microcontrolador adecuado para el producto ya sea para no quedarnos cortos en sus prestaciones o para no desaprovecharlo dejando opciones sin emplear.

El microcontrolador elegido para el proyecto (la elección de este microcontrolador no debe serme atribuida, fue elegido por D. Oliver González Gomeras quien realizo el proyecto predecesor a este) es el DEMO9S08LC60 de la empresa Freescale (antigua MOTOROLA). Debo añadir que la elección de dicho microcontrolador fue la adecuada ya que es de la misma familia del microcontrolador estudiado en la asignatura de Microprocesadores e instrumentación electrónica lo que facilita en gran medida trabajar con el. Las diferencias respecto al estudiado en clase son que incorpora una pantalla lcd por lo que no hay que añadir mas periféricos para el visualizado de la velocidad y otras funciones, además de eso incorpora un puerto mas de entradas y salidas lo cual nos permite un mayor abanico de posibilidades a la hora de conectar diversos sensores o puertos de comunicación.

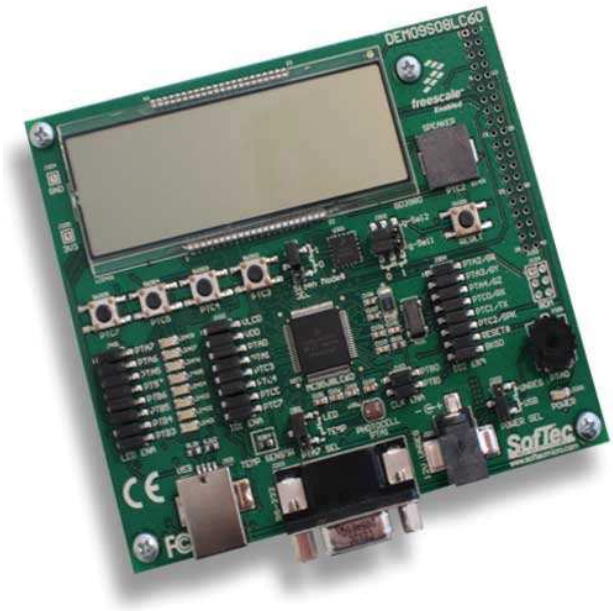


Ilustración 1. Microcontrolador usado en el proyecto.

Imagen cogida del proyecto de Oliver González Gomeras con su permiso

Las funciones que va a desempeñar son las siguientes:

Control de la activación o desactivación del relé de alimentación.

Control de la apertura y cierre de la pinza

Toma de la señal del sensor de velocidad.

Visualización de la velocidad del motor en la pantalla lcd.

Aparte de estas se espera que un futuro próximo también controle la extracción del aire de la campana de vacío, por lo que usaríamos un nuevo sensor y un nuevo relé para desconectar la bomba de vacío. Además también estaría pendiente poner un sensor de nivel de nitrógeno para que controlara una electro válvula. Y finalmente comunicar el microcontrolador con el PC.

Si se desea más información acerca de las justificaciones de la elección de este microcontrolador o de los módulos que posee, véase el proyecto "REGULACIÓN DE VELOCIDAD DE UN PROTOTIPO DE VOLANTE DE INERCIA LEVITADO MEDIANTE SUPERCONDUCTORES" páginas 78 hasta la 83

4. Programar el microcontrolador

Para que realice las funciones requeridas por el usuario el microcontrolador debe ser programado. Leerá las instrucciones que previamente se le han grabado en su memoria interna. Una de las mayores ventajas es la posibilidad de realizar modificaciones en el comportamiento de nuestro proyecto simplemente actualizando el software que ejecuta el microcontrolador.

Como punto de partida comencé con la programación usada en el proyecto antecesor. Siendo el lenguaje ensamblador, además es el estudiado en la asignatura de microprocesadores e instrumentación electrónica por lo que me resultaba familiar, pero enseguida me di cuenta que programar en este lenguaje no resultaba útil ya que el programa se complicaba en gran medida conforme se le añadían mas líneas de código y además cada vez se hacía más difícil de entender para los que no fueran sus programadores, y debía tener en cuenta que detrás de mi otras personas tomaran el relevo y tendrán que añadir líneas a la programación, en función de lo que necesiten. Así que debía buscar un lenguaje de programación que fuera sencillo de entender para los que tengan que modificar lo que yo he programado. Así que entre los muchos lenguajes de programación tales como Ada, Algol, Basic, Clipper, Cobol, Fortran, Java, Pascal...debía decidirme por el adecuado.

El programa suministrado por la empresa Freescale para poder programar el microcontrolador es el Codewarrior el cual dispone la opción de programar en lenguaje ensamblador, en lenguaje C y en lenguaje C++. En lenguaje ensamblador no era viable ya que se complicaba mucho el seguir el funcionamiento del programa y programar sus funciones. Así que nos quedaba el lenguaje C y el C++, entre ellos existen unas pequeñas diferencias pero en conjunto son muy similares y habiendo sido estudiado de forma básica el lenguaje C en la asignatura de microprocesadores e instrumentación electrónica, me decidí por programar en lenguaje C.

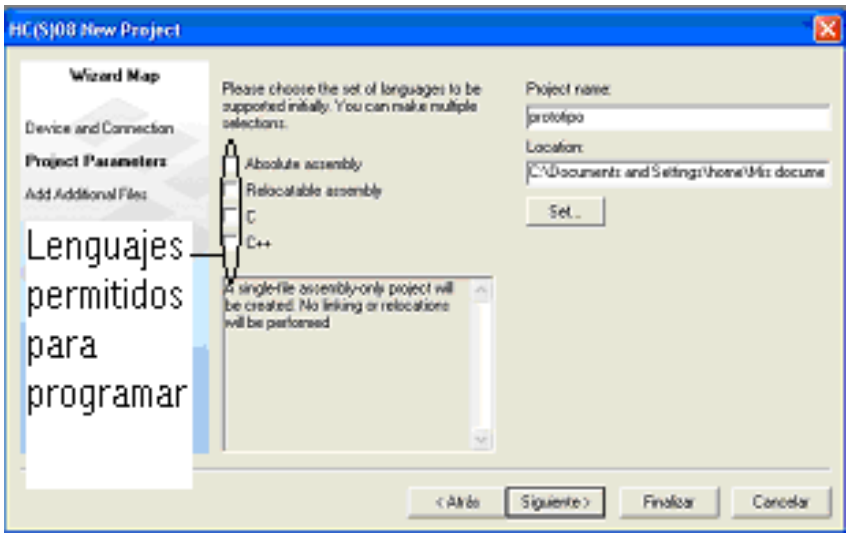


Ilustración 2. *Ventana para seleccionar el lenguaje*

El lenguaje C resulta más entendible ya que en un vistazo podemos hacernos cierta idea de lo que hace dicha función. Además de programar en C use la técnica de

distribuir la programación por rutinas dentro de ellas se efectúan acciones relacionadas por lo que si es necesario modificarlas simplemente hay que ir a dicha rutina y modificarla.

4.1 Diferencias entre ensamblador y C

El lenguaje ensamblador es un lenguaje de bajo nivel cercano al código maquina (código binario). Su gran ventaja frente a otros lenguajes es que ahorra mucha memoria y tiempo de ejecución de cada instrucción.

El lenguaje C usa estructuras de nivel alto lo cual hace que sea más fácil de escribir y de dar mantenimiento.

Veamos un ejemplo de cada uno de ello, ambas son rutinas de retraso.

Programa en ensamblador

```
Retraso10ms     PSHH
PSHX
LDHX #5000
Bucle2          AIX # -1
CPHX #$0000
BNE Bucle2
PULX
PULH
RTS
```

Programa en lenguaje C

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// Delay                                                                                          //
// ---------------------------------------------------------------------------------------------- //
// Retraso de multiplos de 1ms                                                                    //
////////////////////////////////////////////////////////////////////////////////////////////////////

void Delay(unsigned char del)
{
  // Seleccion fBUS como fuente del reloj timer e inicio del timer
  TPM2SC = 0x08;
  while(del)
    if(TPM2SC&0x80)
    {
      del--;
      // limpiar TOF
      TPM2SC &= ~0x80;
    }
  // Parar el timer
  TPM2SC = 0x00;
}
```

4.2 Programación para la aplicación desarrollada en este proyecto

Antes de nada comentar que requerí de dos libros (estos libros aparecen en la bibliografía utilizada son el (1), y el (2)) para hacerme con la programación ya que lo básico visto en clase se me quedaba muy corto para programar las funciones que necesitaba. Además tenía que usar módulos que por falta de tiempo no entraban en el temario de dicha asignatura. Esta falta de conocimientos me complico en el inicio la programación. Mientras la realización de la programación necesite de consultar para resolver mis dudas de las páginas web que aparecen en la bibliografía (9), (11), (12), (13), (14), (15), (16).

A continuación expongo y explico por partes el código grabado en la memoria interna del microcontrolador para su mejor comprensión. Antes de nada decir que las instrucciones del lenguaje C se escriben en ingles. Dentro de la misma programación hay líneas que no son código, estas vienen siempre precedidas por //, simplemente son notas que he puesto para poder ver mejor las partes de las que se estructura el programa, también sirven de recordatorio para no tener que perder tiempo en la búsqueda de las distintas líneas de código y otras son informativas ya que explican en gran medida las líneas de código que veremos tras ellas. Además para facilitar su mantenimiento se ha estructurado todo por subrutinas en cada una de ellas podemos encontrar líneas de código relacionadas con una función.

Lo primero que vemos en la programación corresponde a líneas completamente informativas dicen el nombre del proyecto, autor, fechas, por lo que no son líneas de código solo sirven para que el programador sepa ante que proyecto se encuentra. Por eso a estas líneas se les pone al principio // que significan que lo que se escriba tras ellas son notas del programador las cuales no serán grabadas en la memoria del microcontrolador pero sí que permanecerán en el archivo que crea codewarrior. Tras esto se definen las librerías, esto se hace mediante la directiva #include, las librerías son archivos externos a los que el programa tendrá que llamar por lo que hay que indicarle que las tendrá que tener en cuenta, sin ellas el código que posteriormente escribiremos no sería reconocido como tal y otras definen el microcontrolador que estamos utilizando. En nuestro caso la **hidef.h, derivative.h y lcd.h** controlan ciertos aspectos de nuestro microcontrolador como son los periféricos, la pantalla lcd integrada y funciones internas del microcontrolador para su funcionamiento. La librería **“math.h"**nos permite que las funciones matemáticas sean reconocidas como parte del código. “**Stdio.h"** es la biblioteca estándar del lenguaje de programación C, contiene las definiciones de macros, las constantes, las declaraciones de funciones y la definición de tipos usados por varias operaciones estándar de entrada y salida. **"Stdlib.h"** es la librería estándar de propósito general del lenguaje de programación C. Contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras funciones.

```
////////////////////////////////////////////////////////////////////////////////////////////////
//        prototipo motor con rotor suspendido por campos magnéticos  //
//--------------------------------------------------------------------------------------------//
//                                                                                            //
////////////////////////////////////////////////////////////////////////////////////////////////

#include <hidef.h> /* for EnableInterrupts macro */
#include "derivative.h" /* include peripheral declarations */
#include "lcd.h"
#include "math.h"
#include “stdio.h"
#include "stdlib.h"
```

Una vez definidas las librerías, tenemos que definir las variables que usará el programa mientras se ejecuta. En ellas se irán guardando ciertos datos para que posteriormente otras partes del código las utilicen. Para el proyecto solo he utilizado variables de 2 tipos. Las unsigned short son variables para almacenar datos, números enteros que van desde 0 a 256. Sea elegido este tipo para la variable RPS que son las vueltas por segundo que dará el motor, no es necesario una de tipo int ya que la longitud del dato que guardemos en ella no excederá de 256 que serían 15360 rpm, las unsigned int son variables para almacenar datos, números enteros que van desde 0 a 65535. Para la variable vueltasmin sea utilizado esta última ya que las variables de tipo short se quedaban cortas para guardar este dato. Como su mismo nombre indica la variable vueltasmin guarda las vueltas que el motor efectúa en un minuto. La variable tiempo sirve para almacenar un dato que posteriormente cuando lleguemos a la parte del código donde realiza su función explicare más en profundidad. En esta parte del código además de definir las variables las inicializamos esto significa que le decimos al microcontrolador el valor que tendrán al iniciar el programa. En este caso hemos puesto todas las variables a 0.

```
unsigned short RPS = 0;
unsigned int vueltasmin = 0;
unsigned int tiempo = 0;
unsigned short estado =0;
```

Como anteriormente había comentado la estructura del programa se basa en las rutinas y ésta es la que inicia el programa. Su función es configurar y habilitar todas las funciones que precisará nuestro microprocesador. Para que nuestro programa la encuentre ya que esta rutina será llamada desde el Main (parte principal del programa) debemos darle un nombre y el nombre que le hemos dado es **PeriphInit.** Para que el programa sepa que eso es un nombre de una rutina se usa la instrucción “***void*** *nombre de la rutina* ***(void)”.*** Tras esto comenzamos inicializar los distintos módulos, he añadido notas en las distintas partes para que un futuro programador pueda entender que realiza cada instrucción, he decidido dejar partes en inglés para facilitar su búsqueda en los manuales los cuales están escritos en este idioma. La parte que abarca desde el inicio de la rutina hasta configuración de puertos trata de la configuración del reloj interno, el fijará la velocidad de realización de las instrucciones y de funciones que requieran de referencias de tiempo. La velocidad actual del microcontrolador es de 8Mhz, puede ser modificada y disminuirla pero hay que tener en cuenta que se tendrán que hacer los

cambios pertinentes en otras partes del programa para que los valores de tiempos no se modifiquen pues esos tiempos tienen que ser fijos.

```
////////////////////////////////////////////////////////////////////////////////////////////////
// Inicializacion de los perifericos y registros                                              //
//--------------------------------------------------------------------------------------------//
// los pines que no utilicemos estaran configurados como salida para   //
//que por ellos no entre ruido al micro                                                       //
////////////////////////////////////////////////////////////////////////////////////////////////
 void PeriphInit(void)
 {
 // desactivar COP, habilitar los pines de RESET y BKGD
SOPT1=0x13;

#if!CLOCK
//Seleccion del modo FEE para fBUS=8MHz
//Uso del rango bajo del cristal oscilador.FLL prescaler factor(P)=64
ICGC1=0x38;
// Sets MFD multiplication factor(N)to 4 and RFD divisor(R)to 1
ICGC2=0x20;
//Waita until FLL frequency is locked
while(!(ICGS1 & 0x08));

 #else
// Selects FEI mode
// Sets trimming for fBUS about 8 MHz
ICGTRM = NVICGTRM;
// Uso internal reference clock. FLL prescaler factor (P) = 64
ICGC1 = 0x28;
// Sets MFD multiplication factor (N) to 14 and RFD divisor (R) to 2
ICGC2 = 0x51;
// Waits until FLL frequency is locked
while (!(ICGS1 & 0x08))
;
  #endif
```

Esta parte está incluida en la anterior pero para poder explicarla convenientemente he realizado esta separación.

Lo que tenemos a continuación se trata de configurar las entradas y salidas, en definitiva decirle al microcontrolador por donde estará comunicado con el exterior. Se le dirá por donde recibirá las señales de los sensores y por donde se comunicará con otras partes del control (pinza, relés). Como medida de precaución todos los pines de los puertos que no utilicemos estarán configurados como salida, esto es para evitar que por estos pines entren ruidos electromagnéticos en el microcontrolador, así evitamos posibles errores al tomar valores o datos que pudieran interferir en nuestra programación. Más adelante en otro apartado de la memoria explicaré más detalladamente los pines elegidos para comunicarse entre diversas partes.

```
// configurar puertos

   // PTA[7..0]  output
  PTAD = 0x00;
  PTADD = 0xFF;

  // PTB[7..0]  output
  PTBD = 0x00;
  PTBDD  = 0xFF;
 // configuracion PTC  como output
 // ptc2  entrada del sensor de velocidad en el conector del micro pin 13
  PTCD = 0x00;
  PTCDD = 0b01110011;
//PTC2, PTC7, PTC3, ACTIVO  PULL UP
    PTCPE=0b10001100;
```

Habiendo terminado de configurar los puertos seguimos con la inicialización del modulo TPM y configuración de este para tiempos de 1 milisegundo, mas tarde nos permitirá realizar los retrasos, además de ser necesario para el funcionamiento de la pantalla lcd. El módulo TPM tiene dos canales, en este caso configuraremos para este menester el canal dos.

```
  // Contador en el tpm2 de 1ms
 TPM2MOD = 8000;
  // Stop timer y selección de 1 como prescaler divisor
  TPM2SC = 0x00;
```

Por último en esta rutina inicializo la pantalla lcd y la preparo para su uso posterior. Para concluir cierro la rutina mediante un }

```
 // Initializes LCD Driver Peripheral
 LCDInit();
 LCDOnOffState();
}
```

La rutina siguiente es una rutina especial del TPM1. Aunque este módulo será reconfigurado posteriormente, debe incluirse ahora para que sea reconocido, ya que debe mantenerse un cierto orden en la estructura del código. A este tipo de rutinas se les denomina interrupciones por que se activan cuando ocurre un suceso dejando de ejecutar la parte del código donde se encontraba en ese momento tras ejecutarla vuelve al punto de lectura en el que se encontraba. En este caso se activa cada vez que la bandera del TPM1 se pone a 1, esto ocurre cada vez que el Pin 13 que es el que recibe la señal del sensor de velocidad cambia de 3.51 V a 0 V. Al tratarse de una interrupción del TPM su nombre dentro del programa nos es impuesto por el fabricante (Vtpm1Ch0) y ocupa un lugar especial en el registro de la memoria, el 5. Cada vez que se ejecuta esta interrupción aumentamos en una unidad la variable RPS, tras ser incrementada validamos la interrupción volviendo a poner la bandera en 0 y regresa al código donde se encontraba el programa cuando se produjo la interrupción.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// RUTINA DE ATENCION A LA INTERRUPCION DEL CANAL 0     //
//  DEL       TPM1       (TPM1CH0)                                  //
// SE EJECUTA CADA VEZ QUE HAY UNA VARIACION                  //
//                  EN EL PTC2 DE 0->1                          //
//          LO QUE HACE QUE SE PONGA A 1 EL CHOF                 //
////////////////////////////////////////////////////////////////////////////////////////////////////
interrupt 5 void Vtpm1Ch0(void)
{
  RPS++;     //incremento de la variable

 //  limpia el flag de interrupción CHOF
TPM1C0SC_CH0F=0;
}
```

Esta rutina está sacada de una de las demos que vienen incluidas con el microcontrolador en un CD adjunto al CD del codewarrior, el fabricante nos las facilita para que su comprador mediante esas demos sepa programar convenientemente, en mi opinión es muy beneficiosa esta iniciativa porque permite ver los pequeños detalles de la programación que se pueden pasar por alto, los cuales viéndolos en un ejemplo pueden ser fácilmente subsanados. Tuve que introducirla en el código ya que vi que sin ella la pantalla no funcionaba. No puedo precisar su relación exacta con la LCD pero debe de ser por llamadas a esta rutina desde la librería de la LCD. Me he servido de ella para realizar pequeños retrasos que me permitieran controlar los tiempos de visualización en la LCD. Esta rutina solo tiene lugar cuando desde alguna parte del código es llamada. En este momento es cuando configuramos su duración, teniendo como máximo 255 mseg, ya que usa una variable de tipo char la cual tiene un rango para poder guardar datos en la memoria de 0 a 255.

```
////////////////////////////////////////////////////////////////////////////////
// Delay
// ------------------------------------------------------------------------------
// Retraso de multiplos de 1ms
////////////////////////////////////////////////////////////////////////////////
void Delay(unsigned char del)
{
 // Seleccion fBUS como fuente del reloj timer e inicio del timer
 TPM2SC = 0x08;
 while(del)
  if(TPM2SC&0x80)
  {
    del--;
    // limpiar TOF
    TPM2SC &= ~0x80;
  }
 // Parar el timer
 TPM2SC = 0x00;
}
```

Las dos rutinas siguientes también las he tomado de una de las demo. Ambas están relacionadas con hacer aparecer cadenas de texto en la pantalla.

La primera de ellas llamada PrintString muestra las cadenas de texto de forma fija así que hay un numero de caracteres que podremos mostrar en pantalla, eso está limitado a 7 todos los caracteres que pongamos de más no aparecerán en pantalla, esta limitación se debe a la pantalla la cual no dispone de más segmentos para poder representar las letras.

La segunda se hace llamar SlideString hace que los caracteres de la cadena de texto se deslicen por la pantalla por lo que podremos visualizar cadenas mas largas que con la rutina anterior.

Cada una de estas rutinas es útil para cosas distintas: la primera para la presentación de datos como la velocidad y mostrar en pantalla textos cortos, principalmente a los que hay que atender. La segunda la he usado mas para textos de carácter informativo. Ambas rutinas serán llamadas desde distintas partes del programa para que se ejecuten.

```
////////////////////////////////////////////////////////////////////////////////////////////////
// PrintString                                                                                //
// ------------------------------------------------------------------------------------------ //
// MUESTRA DATOS EN LA LCD                                                                    //
////////////////////////////////////////////////////////////////////////////////////////////////
void PrintString(unsigned char *str)
{
 unsigned char i;
 LCDClear();
 for(i = 0; str[i] != 0; i++)
   LCDPutChar(str[i], i+1);
 }
////////////////////////////////////////////////////////////////////////////
// SlideString
// ----------------------------------------------------------------------------------------
// DESLIZA LAS PALABRAS POR LA LCD
////////////////////////////////////////////////////////////////////////////
 void SlideString(unsigned char *str, unsigned char d)
{
 unsigned char i, j, start;
 for(i = 0; str[i] != 0; i++)
 {
  Delay(d);
  LCDClear();
  if(i < NUM_DIGITS)
    start = 0;
  else
    start = i-NUM_DIGITS+1;
  for(j = start; j <= i; j++)
    LCDPutChar(str[j], NUM_DIGITS-i+j);
 }
 }
```

Esta rutina es del tipo interrupción y se sucede cada segundo, se encarga de hacer posible la visualización en pantalla de la velocidad del rotor y según el valor de la velocidad toma unas decisiones u otras. Como antes he explicado se programó el TPM para que cada segundo produjera una interrupción, ahora nos encontramos en ese punto. Se vió que actualizar la pantalla cada segundo no merecía la pena ya que la velocidad no variaba considerablemente como para realizar la medición de esta solo en este tiempo. Existía la posibilidad de aumentar el TPM pero no mucho más de 1 segundo (ver la rutina de TPM para su explicación) Así que lo que se ha hecho es crear una variable llamada tiempo que se incrementa cada vez que se inicia esta rutina. Si la variable tiempo no es 3 (que serian tres segundos dado que la rutina se produce cada segundo) entonces valida la interrupción y sigue con el resto del programa. En el caso que la variable tiempo sea 3 entra en una subrutina que lo primero que hace es una formula matemática (Esta es la razón por la que se incluyó la librería math) esta fórmula calcula las vueltas por minuto. La variable RPS cada vez que se ejecuta esta rutina va incrementándose conforme le llegan datos por el pin 13 (puede ser que se pierdan algunas vueltas a causa de que necesitamos entrar en esta rutina 3 veces y cada vez que entra al microcontrolador le cuesta algún nanosegundo, si en ese tiempo coincidiera con la llegada de un dato al pin este dato no se registraría no contabilizándose la vuelta, al ser la posibilidad tan baja e incurrir en una pequeña imprecisión no se ha tenido en cuenta).

Uno de los problemas con los que me he encontrado ha sido el visualizar el dato en pantalla. El dato se encuentra guardado en binario y para que el usuario entienda el dato hay que convertirlo a decimal, para ello creé dos rutinas que debían hacer esta labor pero no conseguí que realizaran su propósito, de todas formas las hago constar por si en un futuro se consigue, estas se llaman BinaString e Itoa. Así que me encontraba en la situación de tener el dato pero no poderlo mostrar en pantalla. Además el microcontrolador tomaba las medidas oportunas al producirse el dato que las activaba. Esto se solucionó mediante la instrucción IF la cual hace que si la variable que le ponemos en este caso el dato es igual a la variable entonces que entre en una subrutina que escriba en pantalla mediante la rutina PrintString el dato el cual lo hemos escrito en la programación como una cadena de texto. El problema de usar esto es que tenemos que escribir todos los datos en la memoria del microcontrolador lo cual ocupa mucha memoria y si nos dejamos un dato de escribir este no aparecerá en pantalla siendo ignorado.

Principalmente hay 3 tipos de velocidades las que están por encima del rango, las que están en rango de trabajo y las que están por debajo. Según en qué tipo velocidad haya en el rotor el microcontrolador deberá hacer cosas distintas.

Si la velocidad está por encima del rango de trabajo que es de 2500 el límite superior. Entonces en pantalla nos aparecería una señal de alarma para llamar la atención al operario tras eso se escribe en pantalla +2500 rpm. Tras eso el microcontrolador procede a desconectar la alimentación y a soltar la pinza que sujeta el rotor, se hace en este orden a causa de que si primero se soltara la pinza antes de desconectar la alimentación automáticamente el rotor seria atraído por campo magnético del estator posiblemente produciendo daños debido a la alta velocidad del rotor.

Si la velocidad esta en rango simplemente en la pantalla aparecerá la velocidad a la que se encuentra en ese momento el rotor.

Si la velocidad es inferior a la velocidad de trabajo entonces el microcontrolador muestra una alerta, indica la velocidad a la que se encuentra y procede a sujetar el rotor con la pinza y conectar la alimentación.

Tras mostrar la velocidad en pantalla se limpian las variables RPS y tiempo inicializándolas, además de validar la interrupción y salir de esta.

La variable estado sirve para que la toma de decisión no se esté realizando una y otra vez si ya sea efectuado.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//RUTINA DE MOSTRAR EN PANTALLA Y TOMA DE DECISIONES //
//                    se actualiza cada segundo                    //
////////////////////////////////////////////////////////////////////////////////////////////////////

interrupt 7 void Vtpm1tovf(void)
{
  tiempo++;
  {

  //actualiza pantalla cada 3 segundos asi mejoramos la precision al contar durante mas tiempo. cuando tiempo = 3 entonces hacemos esta parte del codigo y antes de salir de aqui hay que limpiar el tiempo
     if(tiempo == 3){

      vueltasmin=20*RPS;

       {

     if(vueltasmin >= 2300 ){


      LCDWriteSegment(ALARM,0);
      SlideString("+2300 RPM", 255);
       estado = 0;
       PTAD=0b00000000;
       PTAD=0b00000010;
      PrintString("desc de red");

       PrintString("abrir");
           }

    if(vueltasmin < 1700){

     if(estado == 0){
    PTAD=0x00;
    PrintString("cerrar");
     PTAD=0b00000100;
    PrintString("arrancar");
```

```
  estado++;
 }

if (vueltasmin == 0 && vueltasmin < 15 ) {

PrintString("0 RPM");

}
 if(vueltasmin < 37 && vueltasmin > 17) {

  PrintString("30 RPM");
 }

if(vueltasmin < 500 && vueltasmin > 479) {

  PrintString("490 RPM");
}
    if(vueltasmin < 478 && vueltasmin > 459) {

      PrintString("470 RPM");
}

 if(vueltasmin < 468 && vueltasmin > 439) {

  PrintString("450 RPM");
}

 if(vueltasmin < 438 && vueltasmin > 419) {

  PrintString("430 RPM");

}
    if(vueltasmin < 418 && vueltasmin > 409) {

     PrintString("410 RPM");
}

 if(vueltasmin < 408 && vueltasmin > 379) {

  PrintString("390 RPM");
}

 if(vueltasmin < 378 && vueltasmin > 359) {

  PrintString("370 RPM");
}
    if(vueltasmin < 358 && vueltasmin > 299) {

  PrintString("330 RPM");
   }
```

```
  if(vueltasmin < 298 && vueltasmin > 199) {

  PrintString("250 RPM");
   }
    if(vueltasmin < 198 && vueltasmin > 99) {

  PrintString("150 RPM");
}
    if(vueltasmin < 98 && vueltasmin > 79) {

  PrintString("90 RPM");

}
    if(vueltasmin < 78 && vueltasmin > 59) {

  PrintString("70 RPM");
}

 if(vueltasmin < 58 && vueltasmin > 39) {

  PrintString("50 RPM");
}
if(vueltasmin < 998 && vueltasmin > 899) {

      PrintString("950 RPM");
   Delay(255);
  Delay(255);
}

 if(vueltasmin < 898 && vueltasmin > 799) {

  PrintString("850 RPM");

    }
    if(vueltasmin < 798 && vueltasmin > 699) {

      PrintString("750 RPM");
  Delay(255);
  Delay(255);
}
     if(vueltasmin < 698 && vueltasmin > 679) {

  PrintString("690 RPM");
   Delay(255);
  Delay(255);
}
     if(vueltasmin < 678 && vueltasmin > 659) {

  PrintString("670 RPM");
   Delay(255);
```

```
  Delay(255);
}

  if(vueltasmin < 658 && vueltasmin > 639) {

     PrintString("650 RPM");
   Delay(255);
  Delay(255);
}
     if(vueltasmin < 638 && vueltasmin > 619) {

  PrintString("630 RPM");
   Delay(255);
  Delay(255);
    }
       if(vueltasmin < 618 && vueltasmin > 599) {

  PrintString("610 RPM");
   Delay(255);
  Delay(255);
   }
      if(vueltasmin < 598 && vueltasmin > 579) {

  PrintString("590 RPM");
   Delay(255);
  Delay(255);

}
       if(vueltasmin < 578 && vueltasmin > 559) {
     PrintString("570 RPM");
   Delay(255);
  Delay(255);
}
     if(vueltasmin < 558 && vueltasmin > 539) {

     PrintString("550 RPM");
     Delay(255);
  Delay(255);
}
   if(vueltasmin < 538 && vueltasmin > 519) {

    PrintString("530 RPM");
  Delay(255);
  Delay(255);
}
   if(vueltasmin < 518 && vueltasmin >=500 ) {

  PrintString("500 RPM");

     }
```

```
  if(vueltasmin < 1550 && vueltasmin > 1500){

  PrintString("1525 RPM");

}
   if(vueltasmin < 1600 && vueltasmin > 1551) {

  PrintString("1575 RPM");
   }
      if(vueltasmin < 1650 && vueltasmin > 1601) {

  PrintString("1625 RPM");
       }

        if(vueltasmin < 1700 && vueltasmin > 1651) {

  PrintString("1675 RPM");
   }

    if(vueltasmin < 1499 && vueltasmin > 1479) {

  PrintString("1490 RPM");
}

 if(vueltasmin < 1478 && vueltasmin > 1459) {


   PrintString("1470 RPM");

}

  if(vueltasmin < 1458 && vueltasmin > 1439) {


  PrintString("1450 RPM");

}

 if(vueltasmin < 1438 && vueltasmin > 1419) {


  PrintString("1430 RPM");

}

 if(vueltasmin < 1418 && vueltasmin > 1399) {

  PrintString("1410 RPM");
```

```
}

 if(vueltasmin < 1398 && vueltasmin > 1379) {

   PrintString("1390 RPM");
     }

if(vueltasmin < 1378 && vueltasmin > 1359) {

   PrintString("1370 RPM");
    Delay(255);
   Delay(255);

}


 if(vueltasmin < 1358 && vueltasmin > 1339) {


   PrintString("1350 RPM");

}


 if(vueltasmin < 1340 && vueltasmin > 1320) {


   PrintString("1330 RPM");


}

 if(vueltasmin < 1318 && vueltasmin > 1299) {

   PrintString("1310 RPM");


}

if(vueltasmin < 1299 && vueltasmin > 1199) {


   PrintString("1250 RPM");

}

if(vueltasmin < 1198 && vueltasmin > 1099) {


   PrintString("1150 RPM");
```

```
}

if(vueltasmin < 1098 && vueltasmin > 999) {


  PrintString("1050 RPM");

}
}

if(vueltasmin < 1750 && vueltasmin > 1701)
      {

  PrintString("1725 RPM");
   }

       if(vueltasmin < 1799 && vueltasmin > 1751) {

  PrintString("1775 RPM");
   }

     if(vueltasmin < 1850 && vueltasmin > 1800) {

  PrintString("1825 RPM");
   }

    if(vueltasmin < 1899 && vueltasmin > 1851) {

  PrintString("1875 RPM");
   }

      if(vueltasmin < 1950 && vueltasmin > 1900) {

  PrintString("1925 RPM");
   }

     if(vueltasmin < 1999 && vueltasmin > 1951) {

  PrintString("1975 RPM");
   }

      if(vueltasmin < 2050 && vueltasmin > 2000) {

  PrintString("2025 RPM");
   }
     if(vueltasmin < 2099 && vueltasmin > 2051) {

  PrintString("2075 RPM");
   }
```

```
          if(vueltasmin < 2150 && vueltasmin > 2100) {

            PrintString("2125 RPM");
        }

          if(vueltasmin < 2199 && vueltasmin > 2151) {

             PrintString("2175 RPM");
        }

          if(vueltasmin < 2250 && vueltasmin > 2200) {

             PrintString("2225 RPM");
        }
           if(vueltasmin < 2299 && vueltasmin > 2251) {

                PrintString("2275 RPM");
        }

      }

    //limpiar el RPS
    RPS= 0;

   // limpiar el tiempo
   tiempo= 0;
   }


//limpia el flag de interrupcion T0F
TPM1SC_TOF=0;
  }

}
```

Esta rutina es la que configura el TPM1 para que funcione como es necesario. La primera línea se trata de los registros de control, aquí es donde se le dice al microcontrolador que active las interrupciones del TPM que funcione como input capture(sirve para capturar el dato proporcionado por el sensor de velocidad), para medir el tiempo el uso del XCLK (reloj interno del sistema) y que use un prescale de 128 (divide los tiempos del reloj) esta parte nos sirve para poder contar 1 segundo.

En los 2 siguientes registros guardaremos el dato **62500,** en hexadecimal ya que es de 16 bits y el microcontrolador usa los registros de 8 bits lo que nos queda en F828, por lo que en uno de ellos guardamos la parte F8 y en el otro la parte 28. Contar hasta ese número le cuesta 1 segundo, de ahí toda la configuración anterior. Una vez alcanzado dicho numero se produce un overflow (desbordamiento) que activa de la bandera que da paso a la rutina interrupcion **7 Vtpm1tovf** anteriormente descrita.

No podemos aumentar mucho más el tiempo al que se produce la interrupcion porque el limite del dato que podemos guardar es FFFF ya que es el dato mas grande de 16 bits que la memoria de este módulo puede guardar. Y al no ser mucho mas grande a 1 segundo he preferido hacer las cuentas con un número entero antes que usar uno con decimales que complicarian los calculos

Tenemos que configurar el TPM1C0SC para que tenga encuenta los cambios de 0V a 5V que se produciran en la entrada del TPM1CH0 que es el pin 2 del puerto C, tras cada cambio activa la interrupción **5 Vtpm1Ch0**

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//                    configuración del TPM                                                       //
//                TPM1SC: selección del XCLK, Prescale 128                                        //
// config. del MOD a 1 segundo con clock de 8MHz.                                                 //
//Necesitaremos  62500 ciclos para obtener 1 segundo                                              //
//          asi que 16 bits F828                                                                  //
// incremento del contador cuando se produzca un cambio de 0->1//
  //en la entrada ptc2                                                                            //
// configuracion del TPM para medicion de 1 seg para el delay                                     //
  //del deslizamiento de la pantalla                                                              //
  ////////////////////////////////////////////////////////////////////////////////////////////////////
   void initTPM (void){

   TPM1SC=0x57; //registros de control
   TPM1MODH=0xF8; // registro alto para 1 seg
   TPM1MODL=0x28;  // registro bajo para 1 seg
   TPM1C0SC=0x44;//configuración flancos de subida
  }
```

Esta es la segunda rutina que se ejecuta desde el main y muestra lo primero que el lector puede ver en pantalla; es de carácter informativo. Desde esta rutina se hace llamamiento a otra, la SlideString para que muestre en la LCD las cadenas de texto. Las cuales se deslizarán por la pantalla una tras otra.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
// MENSAJE QUE SE MOSTRARA EN PANTALLA NADA //
//MAS ENCENDER EL MICROCONTROLADOR //
////////////////////////////////////////////////////////////////////////////////////////////////////

void mensajebienvenida (void) {
 SlideString("UNIVERSIDAD DE ZARAGOZA", 255);
 SlideString("EUITIZ", 255);
 SlideString("ING TEC IND ESP ELECTRONICA IND", 255);
 SlideString("PROTOTIPO MOTOR", 255);
 SlideString("DE ROTOR LEVITADO", 255);
 SlideString(" CAMPOS MAGNETICOS", 250);
}
```

Esta rutina es la inicialización del KBI (keyboard interrupt) sirve para decir al microcontrolador que se active las interrupciones debidas a teclados o en este caso a los botones que tenemos en la placa del microcontrolador

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//inizializacion del KBI //
////////////////////////////////////////////////////////////////////////////////////////////////////
void kbinit (void) {
KBI2SC = 0b00000110;
/*          !!!!
           !!!+---KBIMOD=KBI DETEC MODE  0=EDGE ONLY
           !!+----KBIE  =KBI INT ENABLE  1=ENABLE
           !+-----KBACK =KBI ACKNOWLEDGE 1=CLR IRQF
           +------KBI FLAG
*/
 KBI2PE = 0b10000000;
 /*       !  !
         !  +-----PIN3 PARA EL APAGADO DE LA MAQUINA
         +----------PIN7 PARA EL ENCENDIDO */
KBI2ES = 0b00000000;
}
```

Esta es una rutina no finalizada a la cual he llamado sinaire, ya que aún no disponemos de la campana de vacío ni de la bomba que nos permitirá crearlo. Pero me parece buena idea dejarla preparada dentro de la estructura del programa para que cuando esté esa parte operativa su autor pueda añadir líneas de código que sirvan para su control. Ahora meramente enciende un led que indicaría que daría la instrucción de encender la bomba de vacío para que esta se encendiera, también esto nos aparece en pantalla al mostrarse un símbolo de frío y con texto que informa de lo que esta sucediendo. Al final del proceso se apagaría la bomba, esto lo indico al apagar el led. El apagado de la bomba se efectuaría al recibir el microcontrolador una señal de un sensor, esta parte habría que implementarla en la programación.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//EXTRACCIÓN DEL AIRE BOMBA DE VACIO                                                              //
////////////////////////////////////////////////////////////////////////////////////////////////////

void sinaire (void){

PTBD=0x10;
LCDWriteSegment(SNOWFLAKE,0);
Delay (255);
SlideString("EXTRAYENDO EL AIRE",255);
PTBD=0x00;
}
```

Esta es otra de las rutinas provisionales, como dejo constancia en las notas que pongo dentro del archivo de programación.

Ahora esta rutina efectúa la siguiente operación una vez en ella: escribe en pantalla encender y espera a que presionemos el botón de on que es el **PTC3** en cuanto es pulsado continua con la lectura de las lineas de codigo.

Esta rutina servirá es para el encendido del motor sin precipitarnos. El motor necesita un tiempo desde que comenzamos a verter el nitrógeno en el deposito superior hasta que enfría las pastillas superconductoras. Hasta ahora se rellenaba constantemente el deposito y a los 15 min. se suponía que las pastillas funcionaban convenientemente. Entonces se procedía al encendido, como ahora queremos que el microcontrolador haga el control completo del prototipo entonces sino se hubiera escrito esta parte de código nada mas encenderlo, estuviera el campo magnético o no, el motor intentaría arrancar lo cual seria seriamente perjudicial para diversas partes. Mediante esta espera mantenemos al microcontrolador en un bucle del que saldrá al presionar el botón de On y podrá seguir ejecutando las ordenes. En un futuro habrá que buscar algún otro sistema que le permita al microcontrolador seguir avanzando en la lectura de las líneas del código sin necesidad de que un operario presione un botón.

```
////////////////////////////////////////////////////////////////////////////////////////////////////
//Rutina que espera que se pulse el botón de on                                                   //
//-------------------------------------------------------------------------------------------------//
//en el futuro es posible que en vez de usar un botón sea                                         //
//el encendido automático al valorar que los superconductores                                     //
// están lo suficientemente fríos para que el rotor se mantenga levitando                         //
// ¿sensor de temperatura? ¿sensor campo magnético?                                               //
////////////////////////////////////////////////////////////////////////////////////////////////////
void encendido (void){

PrintString ("encender");
Delay(250);
if(PTCD_PTCD3 == 0)

 sinaire();

 else
```

```
encendido();//para qeu se mantenga en esta rutina hasta que se pulse el PTC3 boton on

}
```

A diferencia del botón de encendido que solo se espera que sea pulsado en un determinado momento de la programación. El botón de parado debe ser reconocido por el microcontrolador en cualquier fase haciendo que el microcontrolador deje lo que estaba haciendo para dar paso a la rutina de apagado. Para que esto se realizara se tuvo que configurar el KBI anteriormente descrito. En estos momentos esta rutina se activa al ser pulsado el botón conectado al pin 7 del puerto C, se escribe en pantalla la palabra Parar para que el operario sepa que sea aceptado la interrupción, el microcontrolador por medio del pin 14 da la orden de sujetar con la pinza el rotor y después desconecta de la alimentación y tras esto se valida la interrupción y volvemos a la rutina de encendido para proceder a la espera de que el operario haya solucionado el problema que le llevo presionar el botón de parado.

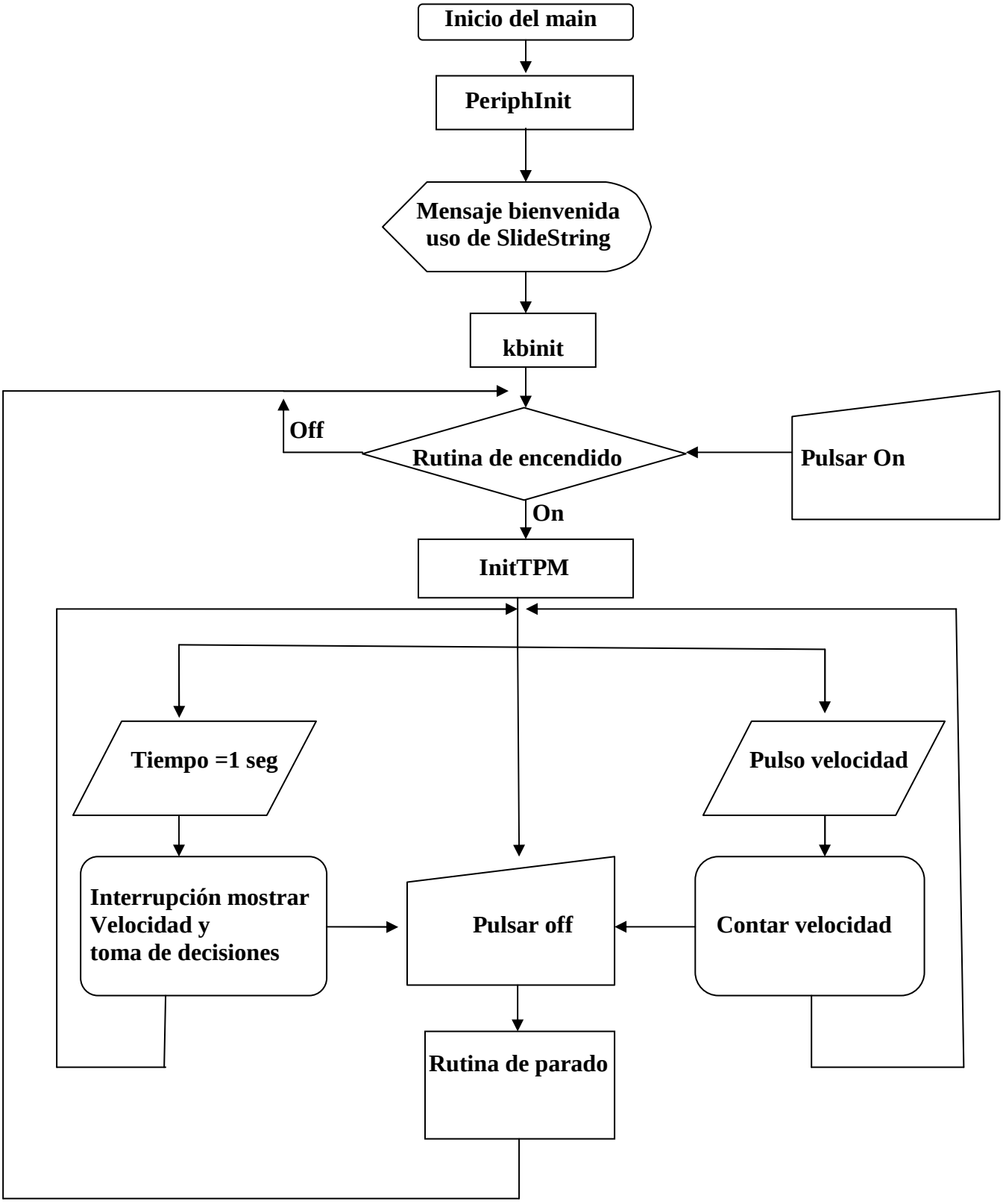
```
////////////////////////////////////////////////////////////////////////////////////////////////////////////
//RUTINA DE INTERRUPCION PARA PARAR EL MOTOR                                                              //
//NECESITAREMOS HACERLO CON EL PULSADO                                                                    //
//                                DEL BOTON PTC7                                                          //
////////////////////////////////////////////////////////////////////////////////////////////////////////////

interrupt 17 void interrupcionkbi (void){


PrintString ("parar");
   Delay(250);
   Delay(250);
  PTAD=0b00000000;;// desconectar de red y cerrar pinza
PrintString("Desc  al.");
PrintString("sujetar pinza");
   KBI2SC_KBACK = 1;
encendido();

}
```

Para finalizar llegamos al MAIN aquí es de donde parte el microcontrolador para efectuar la lectura del código. Aquí introducimos la llamada a las distintas rutinas las cuales tras realizarse vuelve al main y sigue con la siguiente. La llamada de las rutinas sigue un orden adecuado para que nuestro sistema funcione adecuadamente.

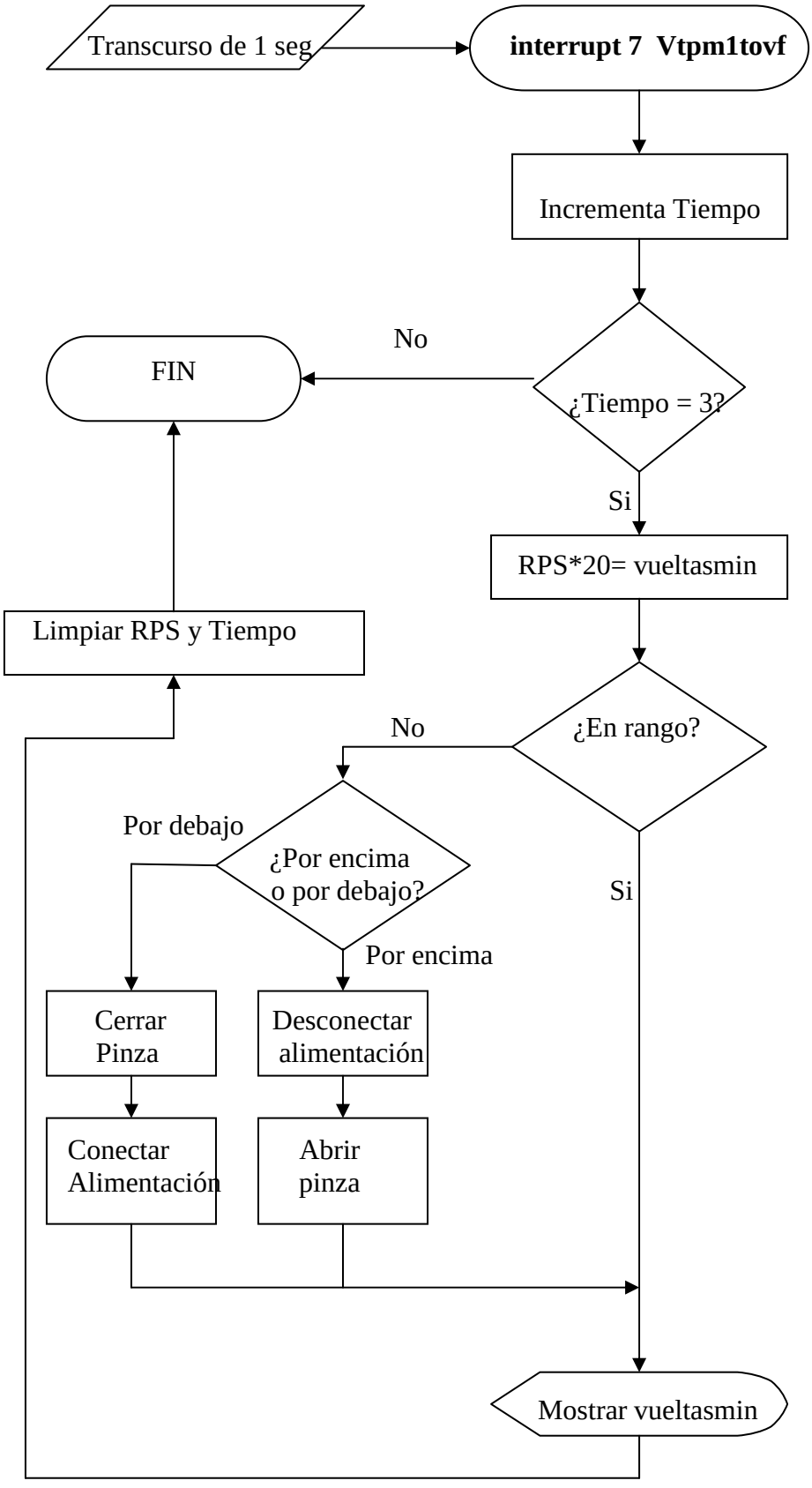
```
////////////////////////////////////////////////////////////////////////////////
// EN ESTA PARTE SE ENCUENTRA EL MAIN                                         //
////////////////////////////////////////////////////////////////////////////////
void main(void)
 {

  EnableInterrupts; /* enable interrupts */
  /* include your code here */
  PeriphInit();
  mensajebienvenida();
  kbinit();
  encendido();

  initTPM();
    //
  for(;;) {

    __RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave main */
}.
```

4.3 Diagramas de flujo

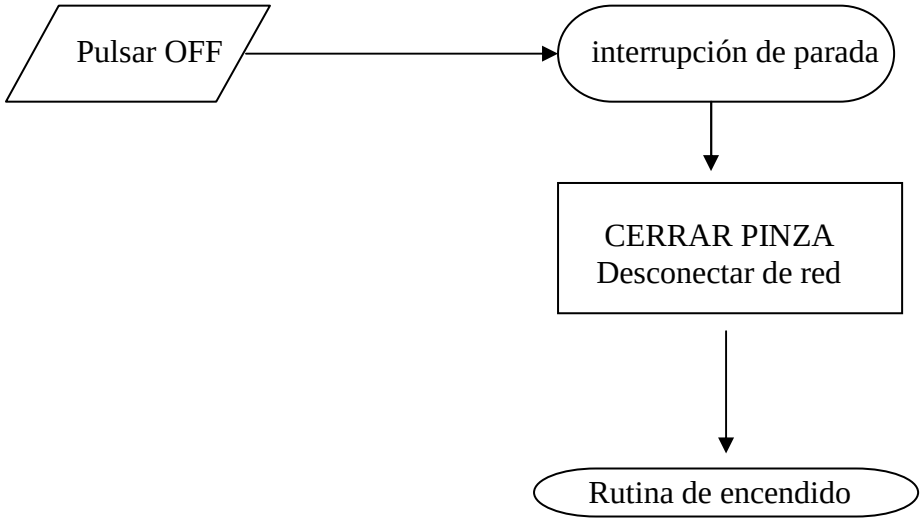
4.3.1 Main



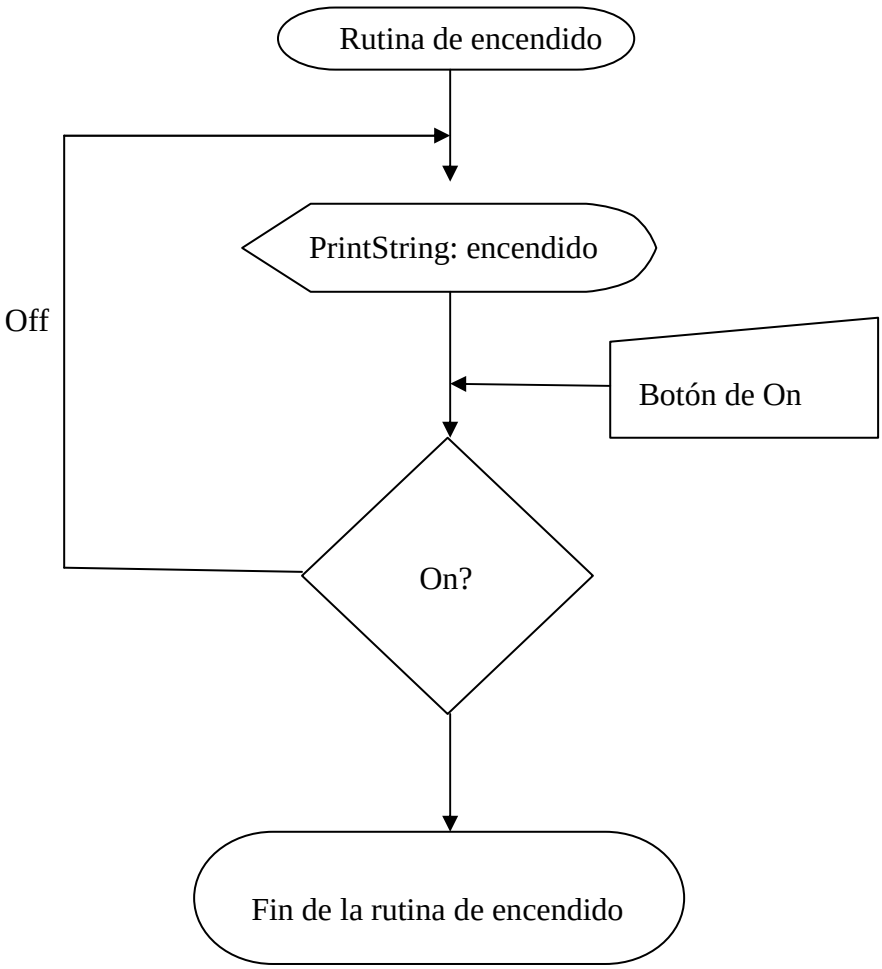
4.3.2 Interrupción del TPM



4.3.3 Interrupción de parada



4.3.4 Rutina de encendido



5. Sensor de velocidad.

Una de las variables a controlar es la velocidad del motor para mantenerlo dentro de un rango de velocidad (entre 1800 y 2500 RPM). Para determinar la velocidad necesitamos utilizar un sensor.

Existen varios sistemas de formas para determinar la velocidad de motores estos son: la dinamo tacométrica, el encoder relativo, el encoder absoluto, sensor de efecto hall y los sensores fotoeléctricos.

El sensor elegido es el fotoeléctrico ya que era el único se adaptaba al proyecto, a continuación expongo los motivos. Tuve que descartar tanto la dinamo tacometrica como los encoders por motivos constructivos del motor, ya que no disponemos de espacio para poder introducirlos, ya que precisan estar acoplados al eje y girar solidarios a este, además nos anula la propiedad fundamental del motor que es no tener rozamiento mecánico, y por tanto perdidas. Se descarto el sensor de efecto Hall por creer que podría verse afectado su funcionamiento ya que el sensor de efecto hall se basa en la creación de un campo eléctrico en un conductor al ser atravesado por un campo magnético. Para ello en el rotor se colocaría un pequeño imán que cada vez que pasara por delante el sensor hall induciría una corriente. Con esto podemos determinar las revoluciones del motor ya que cada vez que se produjeran significaría que el imán a dado una vuelta, pero existía la posibilidad de que su funcionamiento estuviera comprometido por que la zona de trabajo del sensor va a estar muy próxima a uno de los imanes que permite que el rotor este suspendido por lo que el sensor Hall se hallaría bajo la influencia de un campo magnético constante que podría provocar posibles lecturas erróneas. En la siguiente imagen podemos observar la fuerza del campo magnético producida por el imán el cual ha imantado el rodamiento que está en su proximidad y poniendo una llave Allen es capaz de sujetarla.

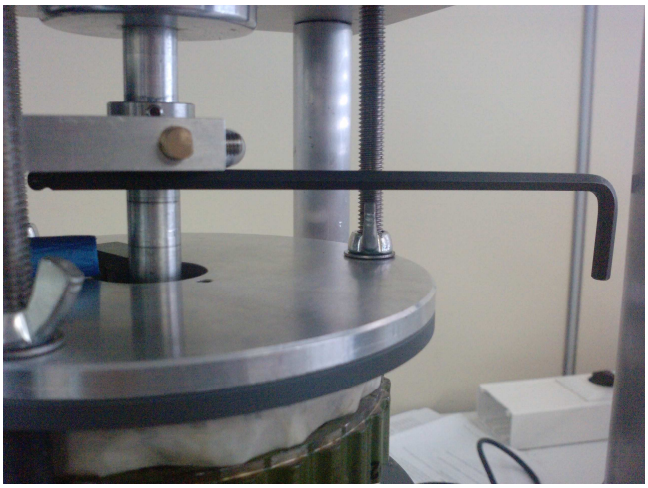
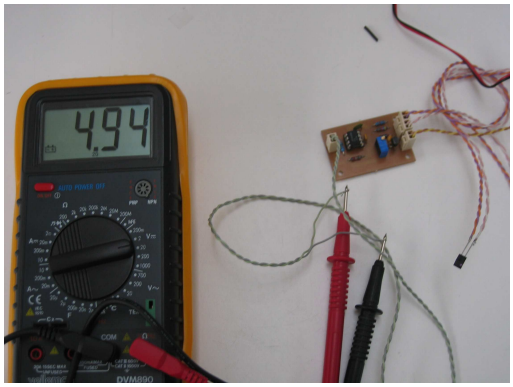


Ilustración 3. *Llave Allen suspendida por el campo magnético*

Entre los tipos de sensores fotoeléctricos decidí que fuera infrarrojo, los hay también verdes, porque es el menos influenciado por la luz ambiente. Aparte del sensor necesitábamos un circuito para acomodar la señal y hacerla mas estable y adaptar los niveles de tensión para poderlo conectar a nuestro microcontrolador, así que se diseño una placa de circuito impreso para esta función. Esta placa fue hecha por el

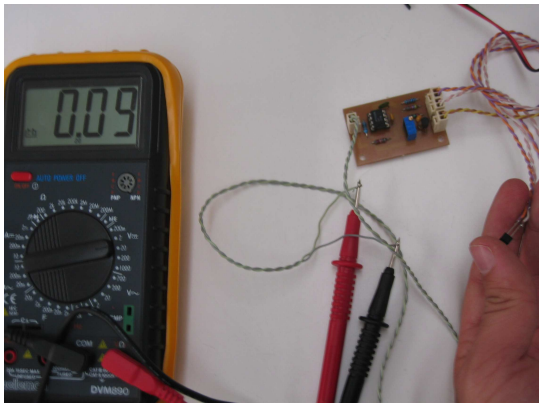
departamento de ciencias. Ellos basándose en el diseño que les facilite y las características que debía tener, hicieron una serie de modificaciones para mejorar el diseño en algún aspecto y para adaptarlo a los componentes a los que tenían mas accesibilidad. El sensor fotoeléctrico utilizado es el OPB706B del fabricante OPTEK Technology. Ver la carpeta datasheet para ver las características del sensor fotoeléctrico, y los planos de la placa impresa del sensor de velocidad pueden verse en la carpeta de documentos.

Antes de conectar el sensor de velocidad al microcontrolador realizamos una serie de ensayos. El primero fue un ensayo estático para ver si los valores esperados en la salida eran los correctos.



Al no recibir reflexión el receptor la salida debería ser de 5 V. Esto es lo que tendría que salir cuando el eje del motor está realizando una vuelta. Según vemos en la imagen el valor está muy próximo a 5 V

Al recibir la reflexión deberíamos obtener el valor de 0 V eso nos indicaría que se a producido una vuelta ya que la reflexión solo se produce cuando el haz de luz choca contra la pegatina que hay puesta en el eje. Como vemos en esta imagen uso mi dedo para que el rayo de luz infrarroja choque e incida sobre el receptor obteniendo un valor muy próximo a 0 V



Después se realizó un ensayo dinámico para realizarlo se utilizo un ventilador de ordenador que nos facilito el maestro de taller D. Joaquín Mur, su finalidad era comprobar que el sensor era capaz de conmutar a distintas velocidades dándonos una señal que después permitiera al microcontrolador contar los flancos (cada flanco una vuelta). Uno de los problemas que podíamos esperar era que el las conmutaciones se produjeran picos antes que la tensión se estabilizara. Estos picos nos falsearían la velocidad induciéndonos a errores, ya que el microcontrolador los tomaría como cambio de flanco. Se trato de observar las conmutaciones usando el osciloscopio pero no se vio ningún pico.

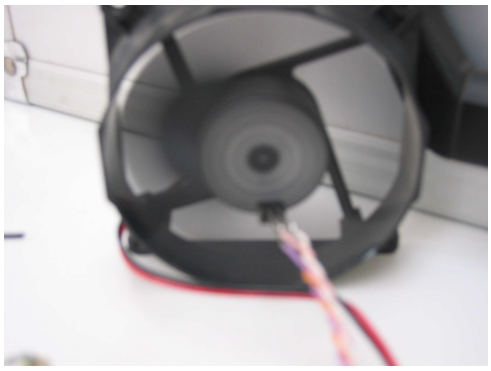


Ilustración 4. *Ventilador usado en los ensayos*

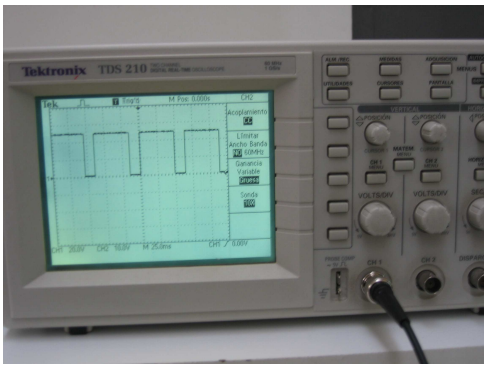


Ilustración 5. *Señal a la salida del sensor*

Aunque según las recomendaciones encontradas en Internet y las referencias del fabricante el microcontrolador podía soportar en la entrada del puerto 5 V. Se decidió bajar a 3,23 voltios para equipararla a la tensión que puede dar el microcontrolador, si el puerto está configurado como salida, la cual es 3.3V. Y proteger en cierta medida al microcontrolador. Así que se realizo un divisor de tensión, colocando una resistencia de 2K1 y otra de 2K9, aproximadamente las caídas tendrían que ser de 2,1V y de 2,9.

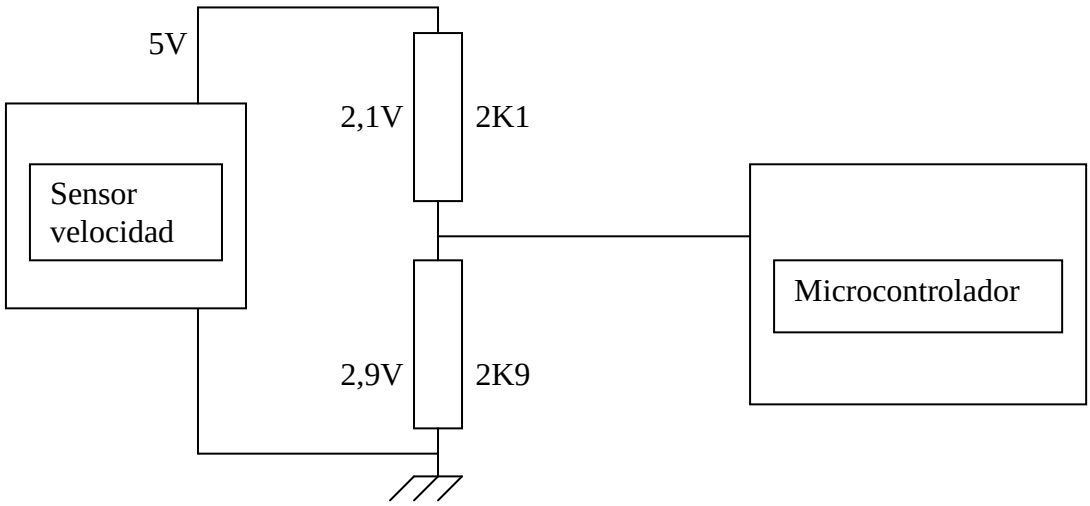


Ilustración 9 *Esquema de conexión microcontrolador sensor de velocidad*

Se observo que las tensiones al estar las resistencias no se repartían como en un principio se había pensado y esto se debió a que no había tenido en cuenta la resistencia interna del sensor la cual se calculo y es de 2KΩ. En circuito abierto el sensor entregaba a la salida 5V pero la tensión total que entregaba con las resistencias era de 3.51V. Se realizaron unos cálculos para conseguir tener a la entrada del microcontrolador una señal con el valor de 3,27 V resistencia que se coloco fue de 3800Ω.

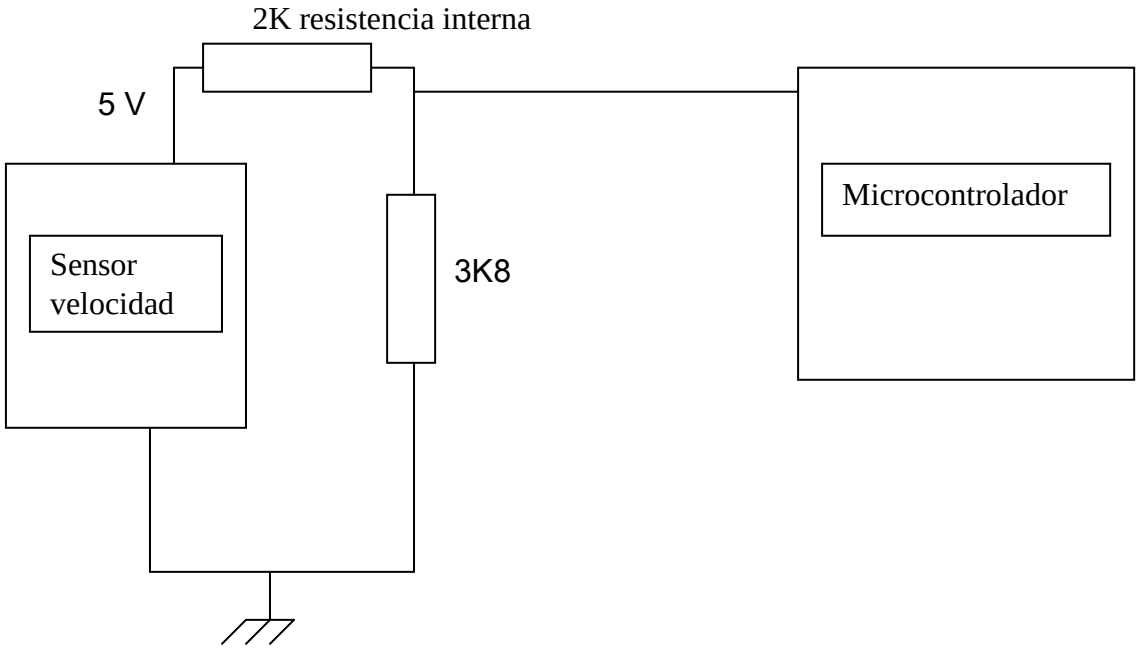


Ilustración 10 *Esquema de conexión utilizado, microcontrolador sensor de velocidad*

Otro de los ensayos una vez conectado el sensor al microcontrolador fue sobre su precisión obteniéndose los siguientes resultados. Se realizo el ensayo con el ventilador anteriormente citado el cual arranca a la velocidad de 550 RPM por lo que fue imposible ver revoluciones por debajo de esta medida ya que este se paraba.

Tacómetro	Microcontrolador
551	520
651	600
662	650
839	800
924	900
1056	1000
1430	1350
1615	1590
1657	1610
1700	1650

Tabla 1.*Velocidades del ventilador registradas por el tacómetro y por el microcontrolador*

Como puede observarse el microcontrolador muestra en pantalla una velocidad que podemos considerar como exacta con algún margen de error.

Nota: Véase en la carpeta planos la placa del sensor, y entre los archivos pdf se encuentra el datasheet del sensor óptico empleado.

6. Sistema de posicionamiento.

Las funciones del sistema de posicionamiento son:

1ª Sujetar el rotor cuando no hay campo magnético, esta situación se da en los periodos en los que no se está trabajando con el prototipo

2ª Se encarga en colocar el rotor a distancia óptima para la levitación.

3ª Cuando el prototipo está conectado a la red impida que el rotor sea atraído por el campo magnético que crea el estator.

Este sistema lo componen distintos elementos: una pinza paralela eléctrica, ella es la que ejercería la fuerza para posicionar el rotor. Para poder transmitir la fuerza de la pinza se realizaron dos piezas en forma de L (los dedos). Los dedos sirven de sujeción de unas piezas llamadas garras que inciden sobre la banda del rodamiento, al cerrarse la pinza las garras gracias a su diseño consiguen levantar el rotor posicionándolo a distancia de levitación, además impiden que cuando el prototipo es alimentado el rotor se mueva de dicha posición. Al rotor se le ha añadido un rodamiento al que se le puso un casquillo para asegurarlo al eje y una banda exterior que es sobre ella las garras ejercerán la fuerza de la pinza.

Este sistema fue una de las cosas que se quedaron pendientes en el proyecto anterior. Todas las piezas metálicas hubo que diseñarlas y fueron fabricadas en el taller de precisión de la universidad de Zaragoza.

6. 1 Pinza paralela eléctrica.

La pinza elegida por Oliver Gonzalez Gomeras autor del anterior proyecto es la GEP1406C de la marca SOMMER AUTOMATIC. Es capaz de ejercer una fuerza de 350 N al cerrarse además posee de sistema de autoretención para no perder la posición en la que se encuentre aun estando desconectada de la alimentación lo que nos es muy útil en los periodos de no utilización del prototipo.

Ilustración 6. *Pinza parte frontal*

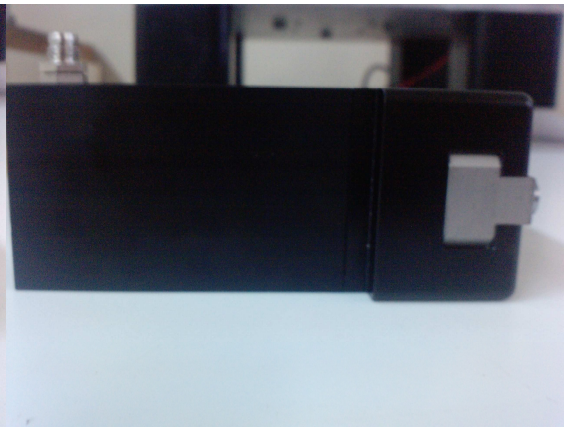


Ilustración 7. *Pinza parte lateral*

Ilustración 8. *Vista superior de la pinza*

Hasta ahora se empleaban estas piezas para colocar el rotor:

Para colocarlo a distancia óptima de levitación: entre las pastillas superconductoras inferiores y el imán del rotor se colocaba una lámina de plástico de pequeño espesor.

Ilustración 9. *Lámina de plástico*

Para impedir que el rotor fuera atraído por el estator: se usaban dos centradores. Colocados arriba y debajo del estator y entre el eje. De esta forma se evitaban los movimientos laterales producidos por el campo magnético.

Ilustración 10. *Centrador*

Cómo funcionaba:

Cuando las pastillas superconductoras ya se encontraban a temperatura optima, esto de determina principalmente viendo que en el depósito superior del nitrógeno comenzamos a tenerlo en estado liquido y en las partes menos aisladas del circuito de refrigeración comenzaban a formarse cristales de hielo. En este momento se retiraba con cuidado la lámina de plástico quedando así el rotor suspendido. Este era momento en el que se podía alimentar el prototipo.

Ilustración 11. *Detalle de la formación de hielo*

Una vez en marcha cuando alcanza velocidades cercanas al rango de trabajo las oscilaciones laterales del rotor se reducen quedando en un estado estable, es este el momento de quitar el centrador superior, esta operación se realiza para disminuir el rozamiento. Para esta operación tenemos que tener sumo cuidado ya que podemos desestabilizar el rotor, el cual esta en movimiento por lo que puede comenzar a rozar en distintas partes pudiéndose dañar. El otro centrador es inviable quitarlo ya que el rotor se desestabilizaría completamente al ser atraído por el estator.

El principal problema de esta operación es el riesgo al quitar el centrador ya que podemos desestabilizar el rotor y provocar rozamientos indeseados que a la larga pueden causas daños a las espiras y tampoco debemos olvidarnos que esta operación se efectúa manualmente con el cierto riesgo hacia el operario y que en un futuro próximo al estar la campana de vacío no tendremos acceso para poder quitarlos y uno de los objetivos es no tener rozamientos. Así que uno de los objetivos de este proyecto era evitar realizar estas operaciones, por eso se instalo una pinza la cual nos permite quitar los centradores y la lamina de plástico.

Para poder instalar la pinza en la estructura del prototipo hubo que hacer una serie de modificaciones, además sobre el eje del rotor debíamos poner un rodamiento que en el que agarraría la pinza.

Lo primero que se tuvo que buscar fue un lugar adecuado para colocar la pinza. En un principio no había ninguno que se adaptara a nuestras expectativas, por los laterales del prototipo las espigas que sujetan el estator impedirían que la pinza se abriera. Por la parte de atrás teníamos otra espiga que impedía el acercamiento de la pinza hacia el rotor y en la parte donde se situaría el operario teníamos la torre de salida del nitrógeno la cual nos impedía el acercamiento de la pinza hacia el eje. Modificar la colación de las espigas quedó descartada por la cantidad de modificaciones que habría

que realizar al prototipo. Pero modificar la posición de la torre de expansión del nitrógeno era posible sin suponer grandes cambios,

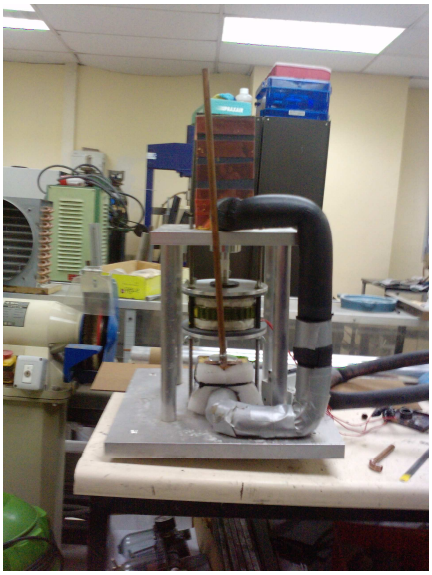


Ilustración 12. *Torre de salida del nitrógeno antes de la modificación*

Así que se procedió a desoldar el codo inferior para añadirle otro tubo de 10 cm. para poder llevar la torre a uno de los laterales. Se añadió un codo más para poder hacer las uniones entre los tubos de cobre. Intenté en primera instancia desoldar con un soldador eléctrico potente pero resulto imposible llevar al estaño de la soldadura al estado líquido ya que el cobre de la tubería rápidamente disipaba el calor y era imposible calentar toda la superficie en su totalidad para que se soltara la unión. La operación de desoldado fue realizada por los maestros de taller del departamento de materiales al ser requerido el soplete oxiacetilénico.

Ilustración 13. *Detalle soldadura codo*

Ilustración 14. *Detalle soldadura*

Ilustración 15. *Nueva posición de la torre*

Las tuberías volvieron a ser aisladas (se reutilizaron los antiguos aislantes) tras comprobar que no había fugas por las nuevas soldaduras.

Ilustración 16. *Torre aislada*

Mientras se realizaba esta operación utilizando el soplete oxiacetilénico dada la proximidad de la tubería de la torre y el estator, se produjo un pequeño accidente y la llama del soplete incidió muy brevemente sobre este quemándose superficialmente el papel protector del estator.

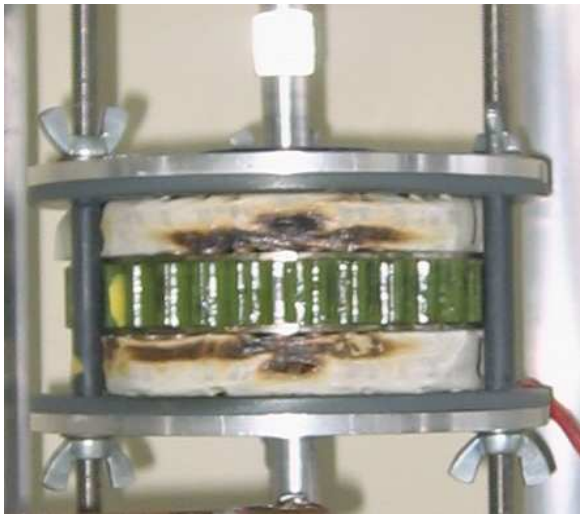


Ilustración 17. *Cinta protectora quemada*

En un primer momento se pensó en que el recubrimiento de las bobinas pudiera haber sufrido algún daño. Así que se realizaron varios ensayos para comprobar el estado de las bobinas y del barniz que las recubre. Uno de ellos fue comprobar si había alguna variación del valor de la resistencia en cada una de las fases. Dicho ensayo nos mostró que el valor en las tres bobinas era idéntico y no existía variación respecto de los valores que se esperaban eso nos daba cierta esperanza de que los daños eran menores de los que podían haber sido. Pero aún quedaba el ensayo de puesta en marcha para comprobar que el barniz aislante mantenía sus propiedades al calentarse las bobinas ya que cabía la posibilidad que el aislamiento en algún punto no fuera del todo el necesario. El maestro de taller D. Joaquín Mur nos facilitó un spray que se usa en motores para recuperar el aislamiento de las bobinas. Se retiro con sumo cuidado el papel quemado para poder acceder a las bobinas y poderles dar el tratamiento

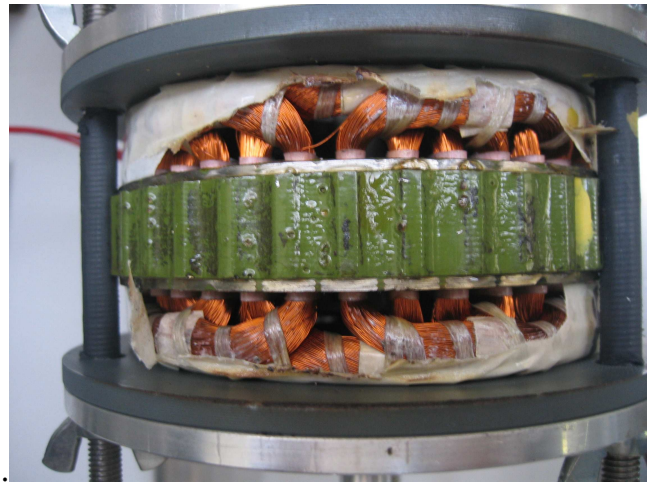


Ilustración 18. *Cinta quemada retirada*

Tras esto se realizó un ensayo de puesta en marcha mediante un variador de frecuencia. Se fue aumentando la tensión del motor progresivamente para ver si se producían chispazos entre las espiras, se llevo el motor hasta rango de trabajo sin que se percibiera ningún problema. Por lo que se llegó a la conclusión que el motor estaba en perfecto estado. Y como última cosa se procedió a tapar las espiras con un nuevo papel protector.

Ilustración 19. *Colocación de la cinta nueva*

6.2 Colocación de la pinza en el prototipo.

Ya resuelto el sitio donde poner la pinza necesitábamos como fijarla a la estructura. Para esta finalidad se diseño esta pieza, la cual se sujetaría a la base superior mediante 4 tornillos M6. Se mando la base superior del prototipo al taller de precisión para que le hicieran 4 agujeros roscados donde se alojarían los tornillos que unirían ambas piezas.

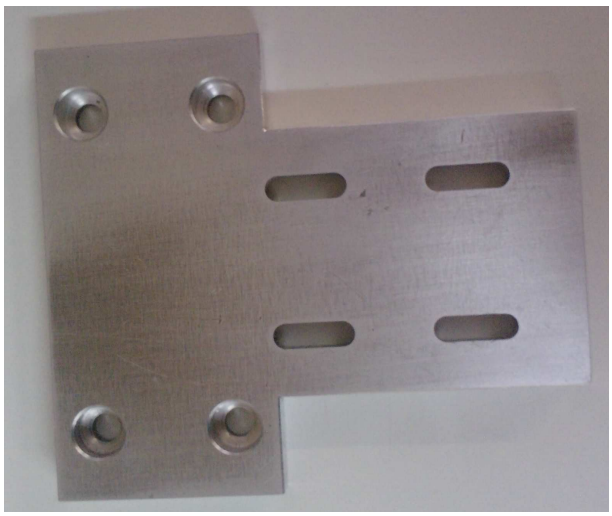


Ilustración 20. *Pieza*

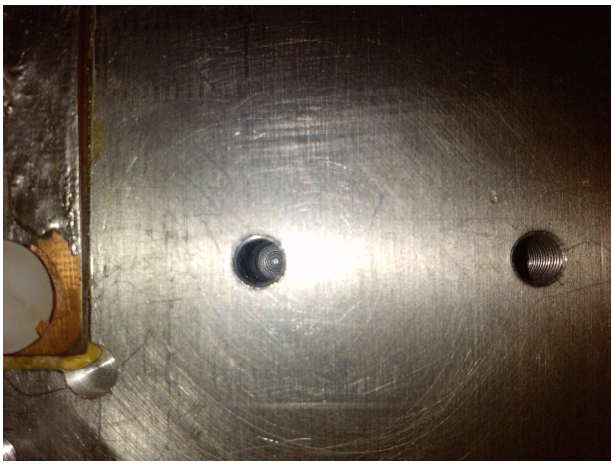


Ilustración 21. *Agujeros roscados realizados en la base superior*

Véase el plano Nº4 para ver las medidas. Una de las cosas que se tuvo en cuenta fueron las medidas de la base inferior, ya se están usando para la construcción de la campana de vacío, por lo que teníamos que adaptarnos a esta limitación ya que de lo contrario la campana cuando se instale no la podríamos encajar.

La pinza se fija a esta pieza mediante 4 espigas roscadas de métrica 5. Las espigas pasan por los agujeros que existen en la pinza los cuales son de dicha métrica nos permiten cierto juego hacia arriba y hacia abajo para poder poner la pinza a la

misma altura que el rodamiento del eje. Para sujetar la pinza en dicha posición se utilizan 8 tuercas y 8 arandelas. Las espigas se sujetan a la pieza mediante otras 8 tuercas que impiden cualquier movimiento. La pieza permite que las espigas puedan mover la pinza hacia adelante y hacia atrás para poder ajustar mejor la distancia de los dedos al centro del eje.

Nota: existe la posibilidad de usar espigas de métrica 6 ya que los agujeros en la pinza están roscados con esa métrica pero con la limitación que solo están en parte haciendo más complicado y limitado el ajuste por eso se descarto. De hecho la pieza por si acaso se hizo para que también pudieran pasar las espigas de métrica 6

6.3 Los dedos

Los dedos son la parte que sirve para transferir la fuerza de la pinza y sustentar las garras. Trato de simplificarse al máximo su diseño. Se usaron las referencias que daba el fabricante de la pinza. La longitud de los dedos se debe a la distancia existente entre la posición de la pinza una vez instalada y el eje del rotor que es donde estará el rodamiento. La fuerza que puede ejercer la pinza depende de la longitud de los dedos los cuales son de 57 cm, así que la fuerza se agarre será de 250N, en las hojas de características de la pinza aparece un grafico de fuerzas donde puede verse pág. 12. Esta fuerza es más que suficiente para la labor que va a desempeñar. Los dedos se sujetan a la pinza mediante 4 tornillos de métrica 6 y 4 casquillos de centraje a los que se les hizo un alojamiento en los dedos para que encajaran. Para sujetar las garras en los dedos hay 1 prisionero tipo Allen de métrica 2.5.

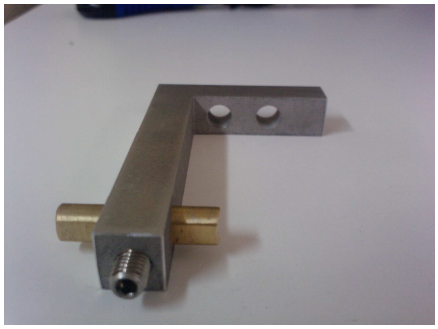
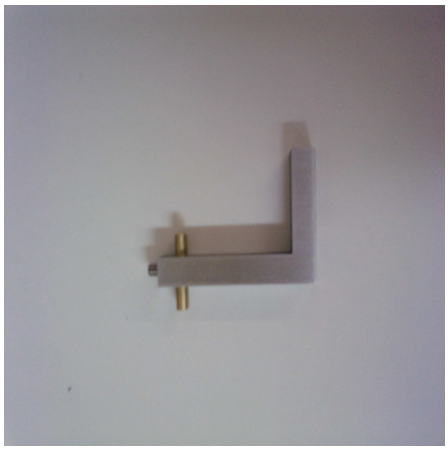


Ilustración 22. *Dedos*

Para ver las medidas ver plano Nº1, podrá encontrar entre los pdfs adjuntos el de las pinzas paralelas eléctricas allí encontrara todas las características de la pinza.

6.4 Las garras

Son las piezas que inciden contra la banda que hay en el rodamiento. A estas piezas se les ha dado una forma cóncava, al contrario que la banda metálica del rodamiento, para que se acoplaran convenientemente, al cerrarse el diseño de convexidad y concavidad de ambas piezas hace que el rotor se levante de la posición que ocupa cuando esta levitando ya que por el peso tiende a caer.



En un primer diseño se creyó que sería suficiente sujetar el rotor por medio de unas garras con tan poca superficie de contacto. Mediante ellas el rotor se sujetaba pero al soltarlo no eran capaces de recuperarlo. Se estuvo intentando ajustarlo al máximo pero no se tuvo éxito además presentaban otro problema y este era que cuando se conectaba el motor a red aún manteniéndolo a frecuencias de 10 Hz para intentar arrancar el motor las pinzas no lograban sujetar el eje el cual era atraído por el estator bloqueándolo al instante. Así que fueron descartadas y se procedió a rediseñarlas.

Ilustración 23. *Primer diseño sujetando el rotor*

El segundo diseño trataba de suplir las limitaciones de su antecesor se aumento la superficie de contacto tanto longitudinalmente, para impedir que fuera atraído en cualquier dirección, como en espesor, se le dio un espesor mayor que el de la banda del rodamiento para facilitar a la vuelta en posición.

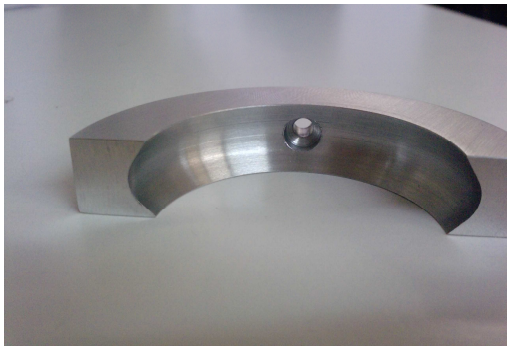


Ilustración 24. *Garras 2º diseño*

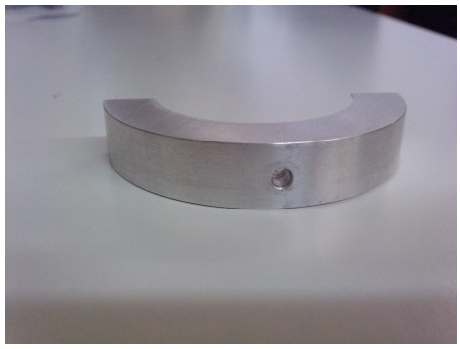


Ilustración 25. *Garras 2º diseño*

Para sujetar la nueva pieza se reutilizo el antiguo diseño al que se le realizo un agujero roscado en el que atornillaríamos un tornillo de tipo Allen.

Ilustración 26 . *Tornillo que sujeta la garra al portagarras*

Ilustración 27. *Portagarras*

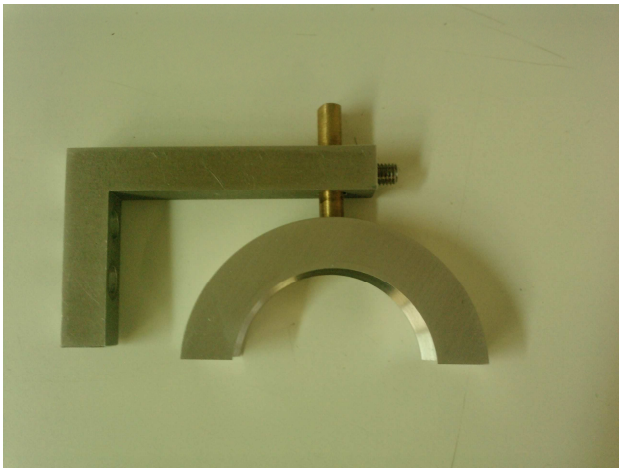


Ilustración 28. *Garra montada*

Una vez instaladas las nuevas garras se presento un pequeño problema, ensayo sin giro en el rotor, y era que al caer el rotor se quedaba levemente suspendido por las 4 esquinas inferiores de las garras. Para solucionarlo se avellanaron las esquinas.

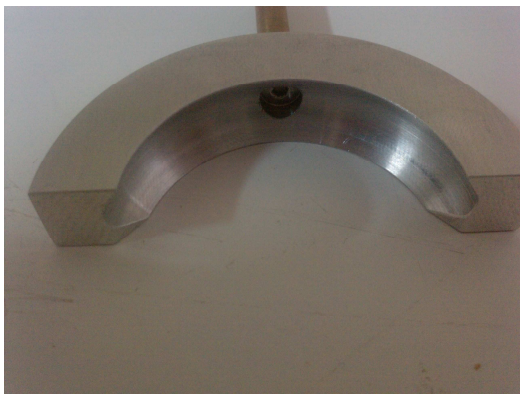


Ilustración 29. *Detalle de la garra con las esquinas avellanadas*

El problema del rotor que se quedaba levemente suspendido de las esquinas fue solucionado con el diseño anterior, pero cuando se probo el prototipo con los superconductores enfriados debidamente y con el rotor girando a 2500 RPM , se vio que el rotor tenía cierta oscilación en su giro y al soltarlo del sistema de posicionamiento este rozaba contra la superficie de las garras, ya que las puntas no se alejaban del rodamiento lo suficiente. Para solucionar esto se cortaron 8 mm de cada lado de las garras.

Para ver las medidas véase el plano Nº2

6.5 El rodamiento

En el eje del rotor se puso un rodamiento al que se le añadió un casquillo para poderlo acoplar ya que el eje no era de una medida normalizada (diámetro de 13,67 mm) por lo que no había ningún rodamiento con esa medida. Se puso un rodamiento con un diámetro interior de 17 mm, al que se le realizó un casquillo. En el casquillo se colocó un prisionero para asegurar el rodamiento al eje para evitar que se soltara a causa de la contracción causada por el frío que produce el nitrógeno liquido que enfría la estructura, en un principio se realizó un casquillo sin prisionero se vio que ajustaba perfectamente hasta que se soltó, atribuyendo este hecho a una diferencia de temperatura que le provocó la contracción del eje y del casquillo. Además de eso al rodamiento se le añadió una banda metálica exterior convexa que seria sobre la que las garras incidirian levantando el rotor para dejarlo en la posición.

6.5.1 La instalación del rodamiento

Para poder poner el rodamiento en el eje hubo que desmontar casi por completo el prototipo se soltaron los 4 tornillos que unen la base superior con las columnas que lo sustentan.

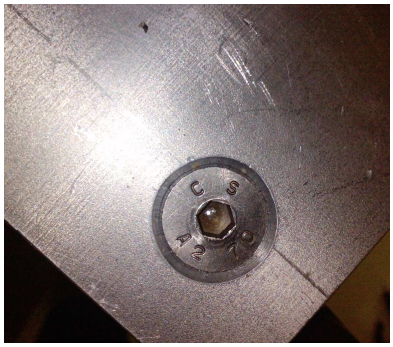


Ilustración 30. *Tornillo*

Antes de retirar la base superior de la estructura hay que soltar el racor que une la tubería que comunica el depósito superior con el inferior.

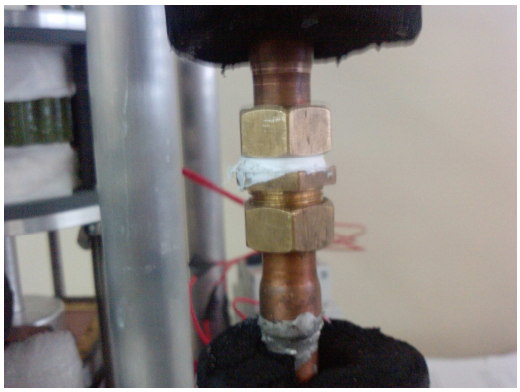


Ilustración 31. *Racor*

En este punto ya se puede retirar la parte superior.

Ilustración 32. *Prototipo desmontado*

Una vez retirada la parte superior soltamos las tuercas mariposas que sujetan la tapa metálica superior que protege el estator ya que el rotor se encuentra en su interior tras esto desatornillamos el imán superior y así de este modo podemos sacar la tapa metálica que anteriormente hemos soltado.

Ilustración 33. *Tuerca mariposa*

Aflojamos el imán inferior para poder retirar el eje con el rotor, al igual que por la parte superior hay una tapa metálica por la parte inferior, así que sino retiramos el imán no se podrá extraer el rotor de dentro del estator. Se soltó el prisionero que sujetaba el rotor al eje.

Se llevó el eje al taller de mecánica de precisión de la Universidad de Zaragoza para que mediante una prensa introdujerán y colocarán en su sitio el rodamiento, el cual quedo completamente ajustado en la parte superior del eje pero aun así se apretó el prisionero para asegurarlo, visto lo ocurrido con su anterior diseño.

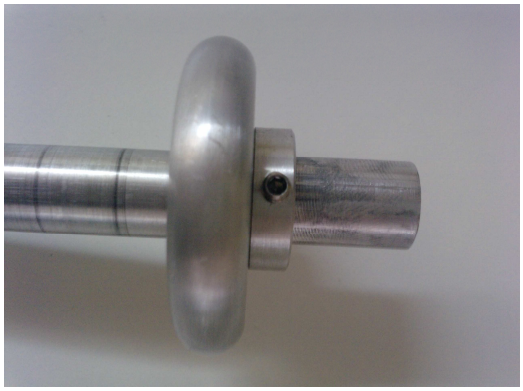


Ilustración 34. *Rodamiento*

Se procedió a realizar la operación inversa para el montaje aunque al añadir el rodamiento se alteró en cierta medida el montaje ahora hay que introducir por la parte inferior del eje la tapa metálica superior que habíamos retirado del estator después meter el rotor y ponerlo en la zona central del eje quedando así la tapa entre el rodamiento y el rotor. Tras realizar esta operación el resto continua siendo igual.

Ilustración 35. *Eje con el rotor, tapa y rodamiento*

Ilustración 36. *Pinza montada en el prototipo*

Una vez ya instalado todo se vio que el rodamiento en eje del rotor le añadía cierto peso de mas que hacia que la distancia de levitación inferior se redujera llegando casi a rozar con el depósito inferior, si se mantenía mucho tiempo sin girar el hielo formado impedía que el rotor volviese a girar. Se estuvo pensando en usar los antiguos porta imanes los cuales eran de plástico y de esta forma reducir el peso ya que los actuales son metálicos pero había muchas reticencias a esta medida ya que fueron eliminados del prototipo en proyectos anteriores a causa de problemas con las dilataciones, al enfriarse las pastillas superconductoras para provocar la levitación. Aun así se cree que en un futuro próximo será necesario reducir el peso del eje haciéndolo de plástico a causa de que con el tiempo las pastillas van perdiendo sus propiedades.

Ilustraciones 37 y 38 . *Vista de la distancia de levitación y formación de cristales de hielo*

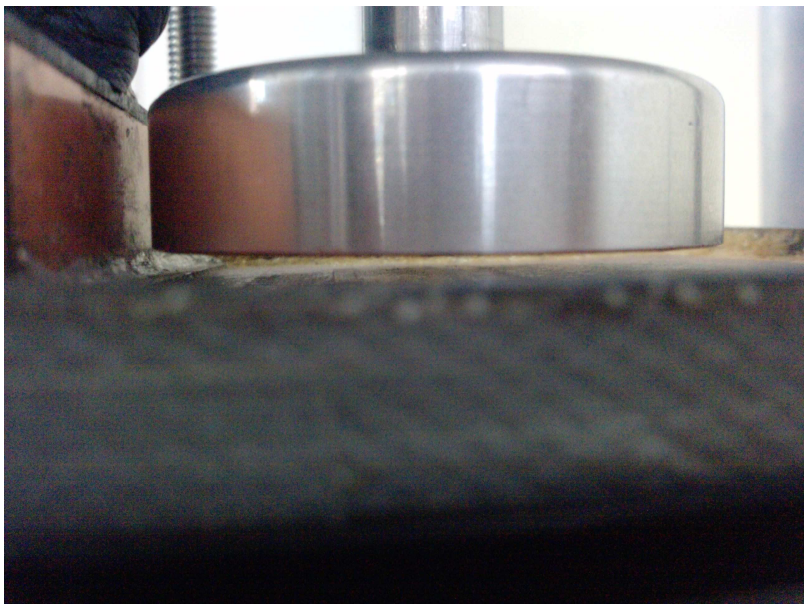


Ilustración 39. *Distancia optima de levitación*

NOTA: cuidado con poner el sistema de posicionamiento lo más arriba posible para conseguir la levitación por que el rodamiento se congela haciendo que no pueda girar el rotor

7. Fuente de alimentación

Hasta ahora para alimentar todo lo anterior se usaba una fuente de alimentación que los maestros de taller del departamento de electricidad me prestaban para realizar los ensayos. Pero no siempre iba ser posible disponer siempre de ella ya que se usa para que los alumnos realicen prácticas.

Ilustración 40. *Fuente de alimentación del laboratorio*

Así que para no tener que depender de su disponibilidad, además de ser grande y teniendo en cuenta que el prototipo será usado para fines didácticos por lo que sería transportado para ser mostrado, se decidió comprar una fuente de alimentación propia más pequeña y que se adaptara mejor a las características que necesitamos de ella. La fuente de alimentación elegida es de la casa MEAN WELL y el modelo es el T-120D dispone de 3 salidas de alimentación a distintas tensiones 24V, 12V y 5V.

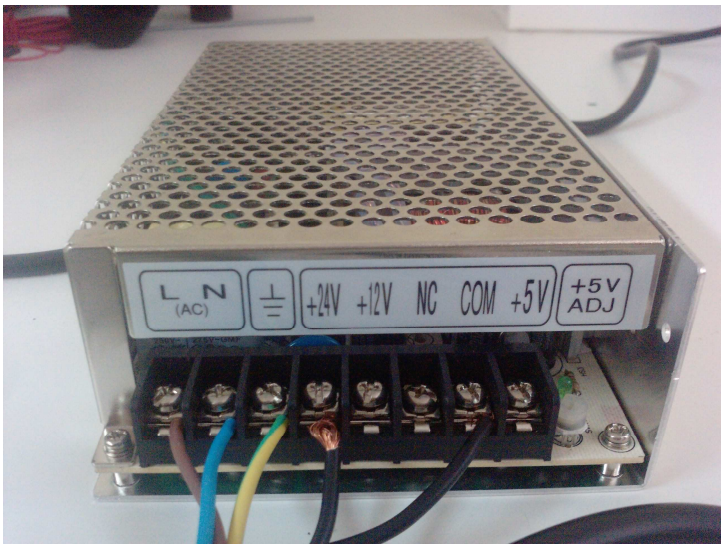


Ilustración 41. *Fuente de alimentación*

Como característica indicar que dispone de un selector de red ya para 220V o 110V

Ilustración 42. *Selector de red*

8.Variador de frecuencia

Hasta ahora se usaba un variador que los maestros de taller del departamento de electricidad me prestaban. Dicho variador, el ALTIVAR 18, era utilizado para la realización de prácticas de la asignatura regulación de maquinas eléctricas así que al igual que la fuente de alimentación no sería posible disponer de él . El variador de frecuencia que se compró es ATV31, uno muy similar al que se empleaba hasta estos momentos. Se mandó al departamento de ciencias para que el añadieran un botón de marcha y otro de paro y lo metieran en una caja de plástico para que estuviera más protegido.

Ilustraciones 43 y 44. *Variador de frecuencia*

Ilustración 45. *Pantalla y controles del variador de frecuencia*

El conector de salida del variador el cual alimenta el motor tuvo que ser soldado, de este proceso destacar lo complicado que resulto al ser el conector demasiado pequeño para poder meter 4 cables y el soldador, y realizar una soldadura buena que no afectara a los otros terminales. Gracias a los consejos del maestro de taller del departamento de electricidad D. Carlos Fuertes se consiguió al poner en cada cable un tubito de goma retráctil por calor para aislar un cable de otro dado el espacio reducido.

En las primeras pruebas con el nuevo variador daba una y otra vez un error, el OPF corte de fase del motor, esto se debía a que el motor es de potencia demasiado baja. Este error se soluciono en variando en el menú FLt el OPL = No. Tras esta variación el motor arranco con normalidad. Para más detalle acudir a la guía del variador en la sección de fallos-causas-soluciones donde aparece completamente explicado.

Además han sido modificados los siguientes parámetros:

ACC =50 seg.
FrS= 130Hz.
HSP=130Hz
FtD=130Hz
Tfr=130Hz
UFt= a L

La razón de modificar el ACC es para que el motor tenga un arranque más suave así evitamos que el estator al comenzar a ser alimentado reciba un pico de corriente que puede provocar descentrar el rotor. Debido a conseguir que el giro del rotor sea más redondo y no tenga cabeceos se ha aumentado la frecuencia de trabajo ya que se observa mayor estabilidad. Para conseguir esto en el variador han sido modificados los siguientes parámetros FrS, HSP, FtD y Tfr. También se ha modificado el Uft que es la elección de la ley tensión frecuencia para conseguir un par constante

9. Conexión de los distintos dispositivos

Hasta este momento he hablado de las distintas partes individualmente en este apartado se muestra como se han conectado entre ellas, y como serán alimentadas. El dispositivo central en torno el cual se conectarán los demás elementos es el microcontrolador. A continuación presento el esquema de conexión:

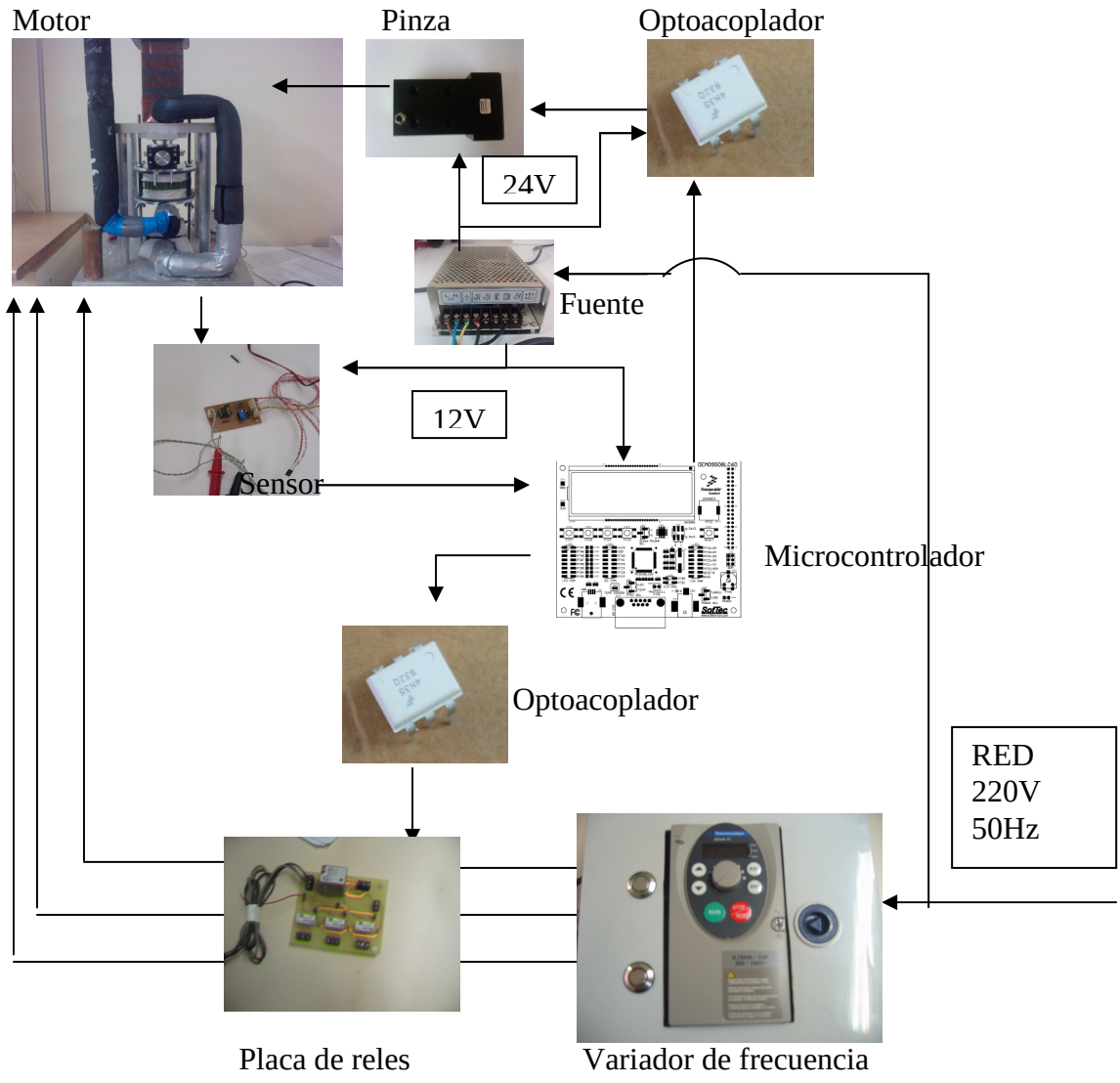


Ilustración 46. Esquema de conexionado de los distintos elementos del prototipo

Aquí podemos apreciar con detalle en qué pines del microcontrolador irán conectados los distintos dispositivos. A la tarea de elegir los pines de conexión hay que prestarle la debida atención porque los pines tienen funciones establecidas.

Nota: para ver las funciones de los pines recomiendo mirar el manual del microcontrolador.

En la parte de programación he comentado que usaremos un botón para que el microcontrolador continué con la siguiente rutina. Aquí indico qué botón deberemos pulsar para dicho propósito.

El botón de off corresponde al PTC7, sea elegido porque esta conectado al modulo KBI que nos permite realizar las interrupciones.

El sensor de velocidad va conectado al pin 13 del conector. Como anteriormente he comentado este pin corresponde al modulo TPM1 en concreto con el canal 0.

El microcontrolador puede estar alimentado mediante una fuente de alimentación de 12 V o mediante el USB. Debemos prestar cierta atención al jumper de selección de alimentación para ver cual está habilitada. La entrada de USB además de ser de alimentación es la entrada para programar el microcontrolador al conectarlo al PC y usar el programa CODEWARRIOR

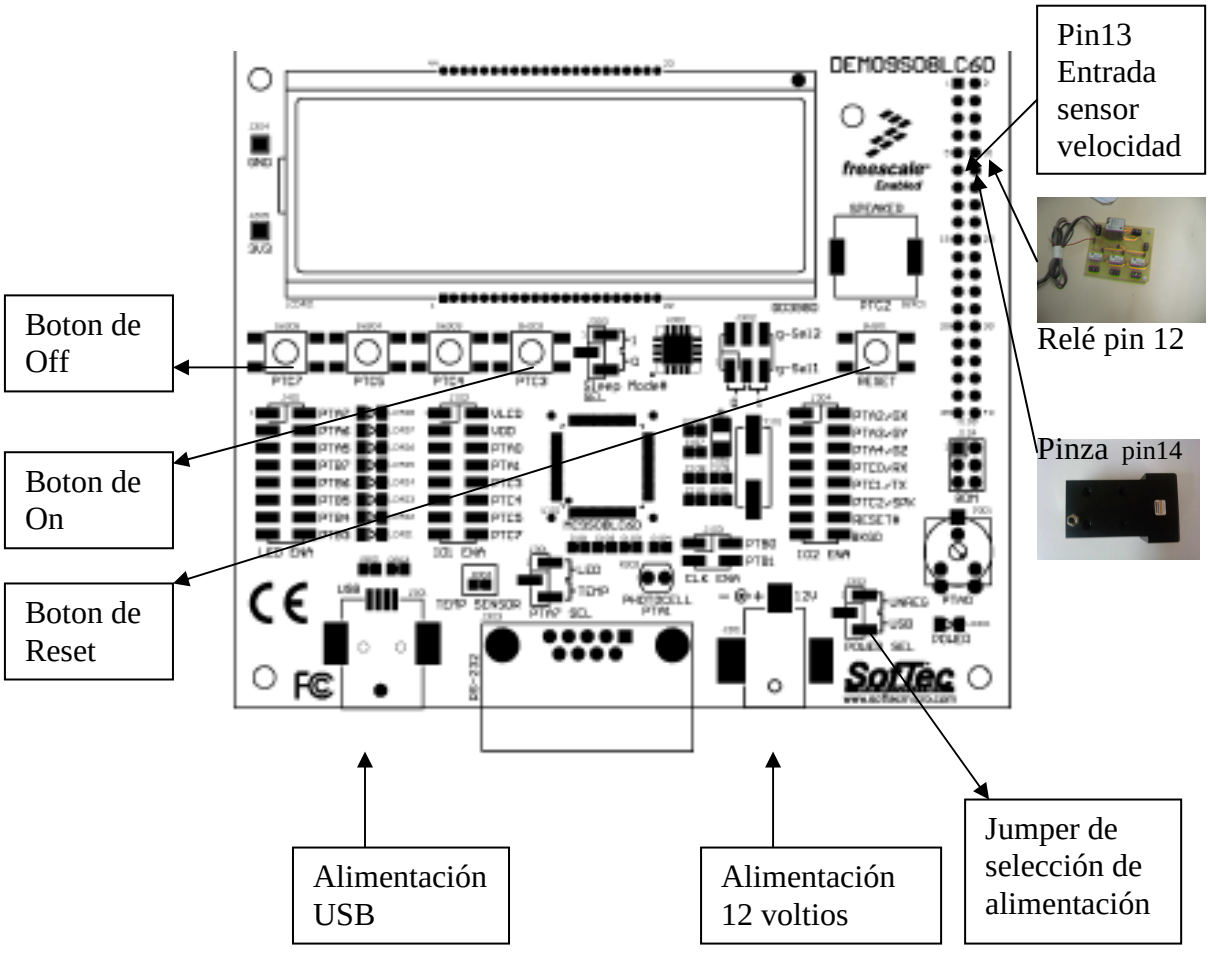


Ilustración 47. *Detalle de los pines de conexión del microcontrolador*

Una vez instalado el sensor en el prototipo se vio que no funcionaba, el microcontrolador no conseguía contar las vueltas del rotor. Se pensó en las posibles causas, en el rotor se le había pegado una cinta reflectante igual a la que se uso en el ventilador del ordenador con el que se hicieron las pruebas. La única diferencia que se encontró fue que el color del eje no era negro. En el proyecto anterior se vio que tuvieron que poner una pegatina reflectante que cubría media longitud de circunferencia del rotor y se puso cinta aislante de color negro sobre la que pegaron la cinta reflectante.

Esta cinta aislante y la cinta reflectante del proyecto anterior fueron retiradas al añadir el rodamiento en el eje. Así viendo que la causa del mal funcionamiento se debía a eso, se pegaron ambas cintas para dejarlo como anteriormente estaba. Solucionándose de esta forma nuestros problemas.

A continuación explicaré el conexionado de la pinza. La cual es alimentada a 24V, sea observado que también funciona siendo alimentada hasta los 12V pero no ejerce la presión máxima la cual necesitamos para evitar que el rotor se mueva de su posición. El terminal de alimentación es el +UB aquí se conectaran 24 Voltios. El –UB es donde conectaremos el terminal COM de la fuente alimentación. EN sirve para indicar a la pinza que este preparada para recibir datos, en nuestro caso siempre lo tendremos en estado lógico 1, para que esté preparado para recibir dato. El D sirve para abrir o cerrar la pinza según el dato lógico que reciba. Si EN no esta habilitado la pinza no modifica el estado en el que se encuentre ya sea abierta o cerrada aunque le estemos metiendo el dato lógico de realizar una de sus posiciones. Los datos lógicos para la pinza son 24 V equivale a un 1 y 0V es un 0.

EN	D
0	0
0	1
1	0
1	1

Si EN se encuentra en el estado lógico 0 es como si se encontrara en estado abierto así que por este terminal circula 0 A además la corriente que circula por el Terminal D circula 0A dando igual que se encuentre en estado 1 como en estado lógico 0. Teniendo la pinza ya fuera cerrada o abierta.

Si EN se encuentra en el estado lógico 1 es como si se encontrara en circuito cerrado por el terminal EN circula una corriente de

Los terminales M+, y M- son los que transmiten las instrucciones para que la pinza se abra o se cierre. El cable para hacer esta unión es suministrado por el fabricante, consta de 3 hilos, azul para el M-, marrón para el M+ y hay un tercer hilo de color negro que no se conecta.

Hay dos terminales más que son el P1 y P2 que según la hoja facilitada por el fabricante deben ir conectados a 0V, pero si hacemos eso sea observado que los led de estado de posición no se encienden, si los conectamos a tierra entonces los dos led aparecen encendidos y no se apagan, mientras que si los dejamos al aire los led se encienden marcando convenientemente el estado en el que se encuentra la pinza en ese momento, así que sea decidido por esta situación frente a los otras dos.

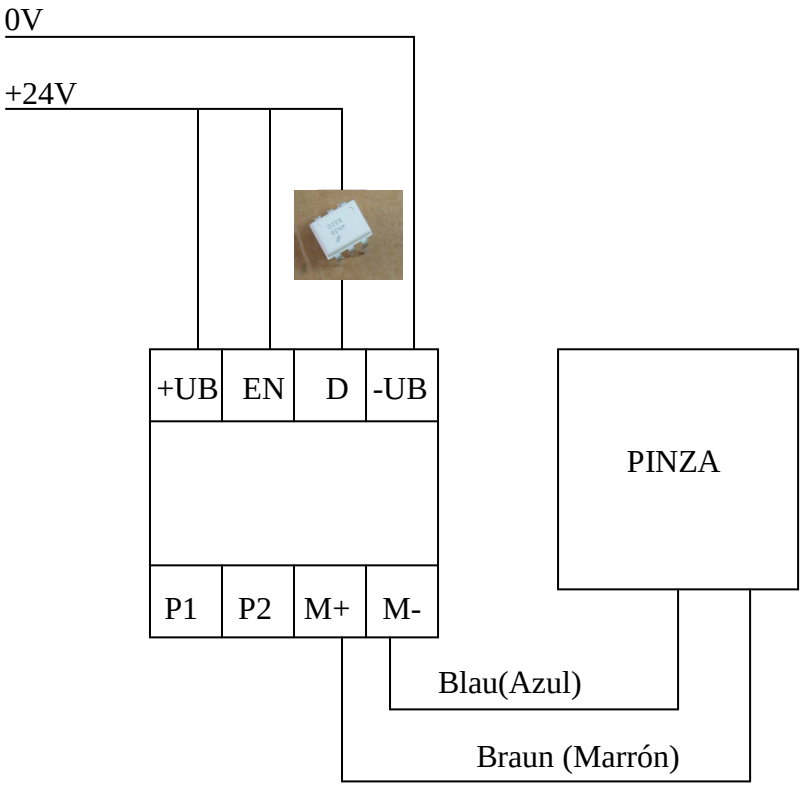


Ilustración 48. *Esquema de conexiones pinza*

Anteriormente he comentado que un 1 lógico de la pinza es 24V mientras que el 1 lógico del microcontrolador es de 3,3V así que era imposible conectarlo directamente. Se uso un optoacoplador para que nos permitiera su interconexión, un optoacoplador funciona como un interruptor excitado mediante la luz emitida por un diodo LED que satura un fototransistor o un fototriac, así la conexión entre ambos es óptica. Tenemos un interruptor el cual aísla eléctricamente la circuitería de la pinza que está a 24 V del microcontrolador que es el que lo controla que esta a 3,3 V. Se pensó en usar un transistor que podría hacer la misma función pero en caso de que el transistor se quemara por diversos problemas entrarían 24V en el microcontrolador dañándolo irremediablemente. Tras valorar las ventajas que ofrecía el optoacoplador frente a su precio de 0,12 € se vio que merecía la pena. El optoacoplador utilizado es el 4N35, es muy común su uso por lo que es fácil de conseguir.

En el ánodo (pin 1) conectamos una resistencia de 1K para limitar la corriente que circula por el diodo, por el otro lado de la resistencia lo introducimos en un pin del microcontrolador. El cátodo (pin 2) lo conectamos a tierra. El colector (pin 5) lo conectamos a 24V, se hizo un ensayo con la pinza para ver la intensidad que circulaba cuando esta en 1 lógico se vio que era 3 mA esta intensidad no suponía ningún problema al optoacoplador el cual podía soportar 100 mA. Por esa razón se decidió no añadir ninguna resistencia para limitar la intensidad ya que no era necesario. El emisor (pin 4) lo conectaríamos a D en la pinza. Los pines 3 y 6 los dejamos al aire.

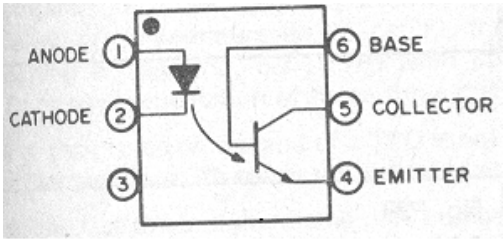


Ilustración 49. *Esquema del 4N35*

Para cortar el suministro de alimentación del motor se reutilizo la placa de relees realizada en el proyecto anterior para proteger el microcontrolador se uso un optoacoplador para su control.

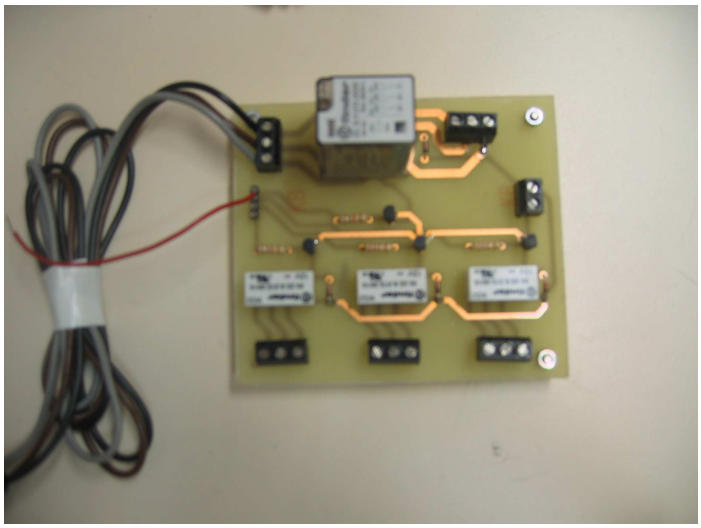


Ilustración 50. *Placa de relés*

El pin 1 lo conectamos al microcontrolador y el pin 2 a tierra. Al pin 5 lo conectamos a 5V que nos da la fuente de alimentación y en el pin 4 conectamos una resistencia para limitar el paso de corriente ya que la placa de relees se diseño para que el mismo microcontrolador se pudiera conectar directamente y su control fuera mediante un transistor por lo que la resistencia que se puso a la entrada del transistor era la adecuada para limitar la corriente a la tensión de 3,3V. la resistencia que hay es de 1KΩ.

$$I=3.3/1000=0,0033\ A$$

Así que con 5V debíamos tener la misma intensidad para no dañar el transistor.

$$R=5/0,0033=1515,15\Omega$$

La resistencia que se obtenía era de 1515Ω y como ya teníamos una de 1000Ω habría que añadir una de 500Ω. Finalmente se añadió una de 1KΩ ya que se comprobó que disparaba igualmente.

$$I=5/2000=0{,}0025\ A$$

2,5mA es la corriente que circula por el transistor y por el optoacoplador cuando el microontrolador da la orden de cerrar el relee de alimentación.

Se realizo una placa de circuito impreso, para la circuitería de control y la acomodación del sensor de velocidad, usando unas placas en las que las pistas ya están trazadas y simplemente hay que cortar mediante un cúter o similar para retirar la pista que no se emplea y para separar los distintos componentes.

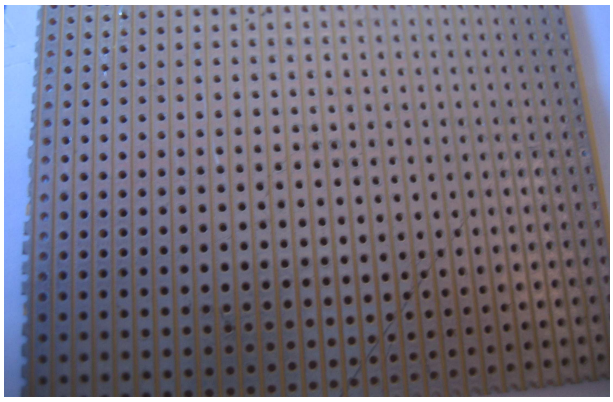


Ilustración 54 *placa de circuito impreso con pistas trazadas*

El plano del circuito impreso es el siguiente:

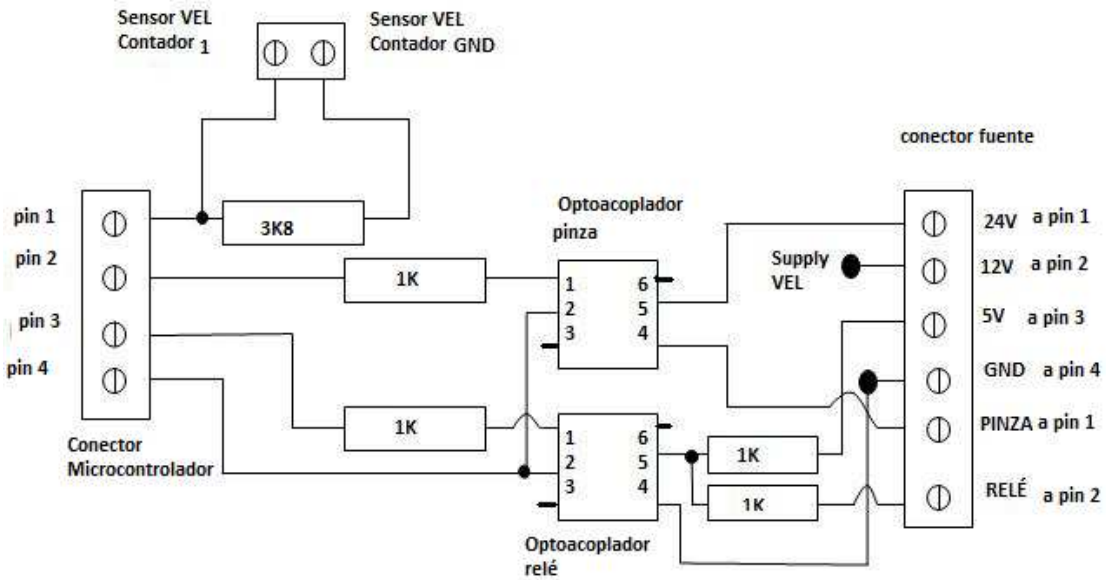


Ilustración 55 *esquema de la placa de acomodación*

Todas las resistencias empleados para limitar la corriente que circulara por los optoacopladores son de 1 K las cuales permiten que las intensidades que circulan se mantengan dentro de los márgenes que pueden soportar dejando un margen de seguridad para evitar posibles daños. En esta placa también se encuentra la resistencia que se ha añadido al sensor de velocidad.

9.1 Ruido electromagnético:

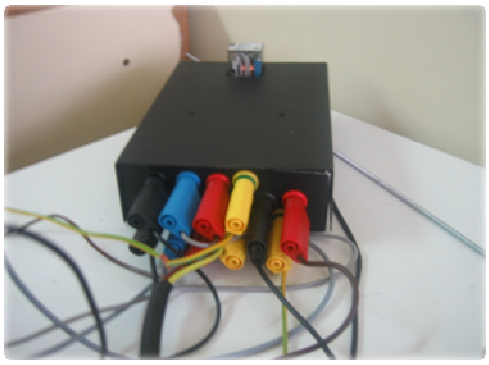
Primeramente para comprobar su funcionamiento se probo todo con el ventilador y todo funcionaba, pero al instalarlo en el prototipo dejo de funcionar correctamente. Al conectar a red el prototipo el microcontrolador se reiniciaba o recibía señales erróneas del sensor de velocidad que le hacían tomar decisiones incorrectas. Esto se produjo a causa del campo magnético que generaban las bobinas del estator.

Este ruido de carácter electromagnético se filtraba a través de los cables y de los circuitos impresos. A 1 metro de distancia el ruido electromagnético comenzaba a reducirse comprobándose en que el microcontrolador realizaba menos decisiones erróneas. Pero para que el sensor de velocidad pudiera tomar las vueltas del rotor el sensor fotoeléctrico debía estar cerca del rotor y por lo tanto cerca del estator, el cual corrompía la señal. Así que hubo que apantallar y proteger toda la circuitería electrónica.

Se compraron cables apantallados que sustituyeron a los que se estaban empleando. Los circuitos impresos se metieron en cajas metálicas. Se compraron conectores metálicos para ponerlos en las cajas, ya que los de plástico no aislaban el sistema de los ruidos electromagnéticos. A las cajas se les puso tomas de tierra, al prototipo se le puso una toma de tierra para reducir el ruido. Con todas estas medidas se consiguió que el sistema funcionara.

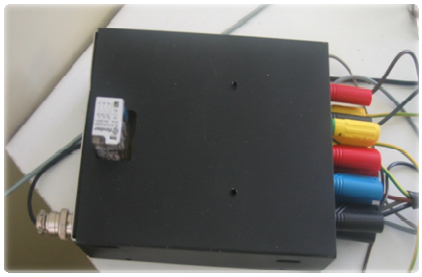
CAJA RELÉ

En su interior se encuentra el relé que conecta o desconecta de red el prototipo



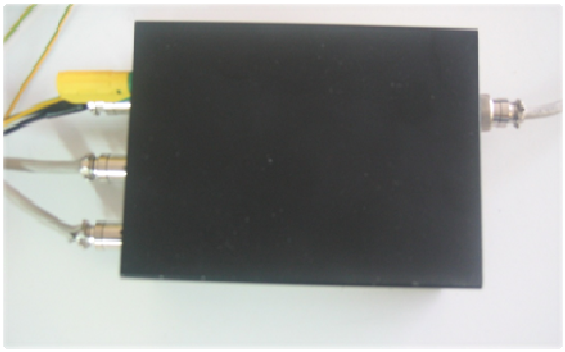
A esta caja se le han puesto conectores de seguridad donde se conectan las bananas de seguridad puestas en los cables del variador y los que alimentan las bobinas del prototipo. La elección de estos conectores se debe a tratar de evitar posibles accidentes. Alimentan al prototipo a 220V. También se ha puesto una toma tierra para el prototipo. El relé es alimentado a 12V para que pueda abrir o cerrar y a la caja tiene toma tierra para apantallar el interior

Por el otro lado de la zona de tensión se encuentra un conector metálico de 2 pines. El pin 2 es el que se encarga en mandar la orden al relé para su apertura o cierre. El pin 1 actualmente no tiene función



CAJA DE ACOMODACIÓN:

En el interior de esta caja se encuentra la placa del circuito impreso del sensor de velocidad y la placa de acomodación.

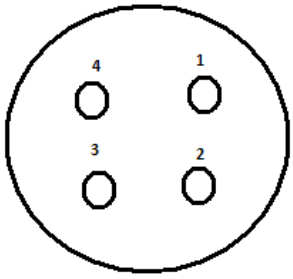


Esta caja también tiene toma de tierra para evitar los ruidos electromagnéticos.

Justo debajo de la toma de tierra se encuentra el conector que permite que la pinza y el relé reciban la señal de control.

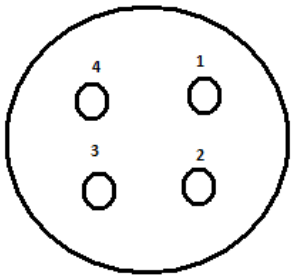
En el pin 1 está conectado el cable que comunica a la pinza y en el pin 2 está conectado al que comunica con el relé. Internamente este conector se conecta con el conector fuente de la placa de acomodación (ver plano circuito ilustración 55 para ver conexión)

El conector que esta junto al anterior es el que alimenta con la fuente de alimentación.



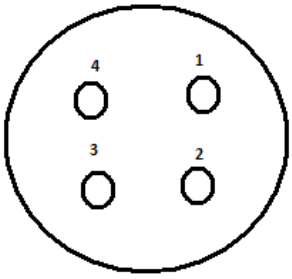
1: Conectado a GND
2: Conectado a 12V
3: Conectado a 24V
4: conectado a 5V

Junto a este está el que conecta el sensor de velocidad.



1 emisor K
2 emisor A
3 receptor E
4 receptor C

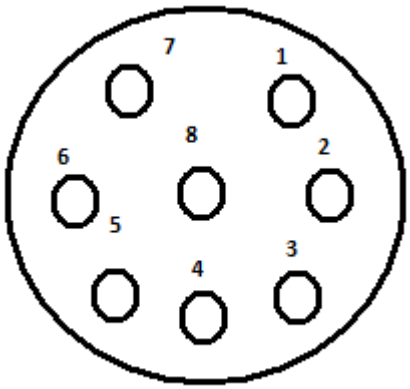
En el otro lado de la caja se encuentra el que conecta con la caja del microcontrolador.



1pinza
2 sensor vel
3 GND
4 Rele

CAJA DEL MICROCONTROLADOR:

En su interior se encuentra el microcontrolador. Se le han añadido botones a la caja para que puedan pulsarse los botones de reset, on, off. Para poder visualizar la pantalla se ha realizado una ranura a la que sea puesto una lamina de plástico para evitar que entre polvo al interior. El conector instalado en esta caja es de 8 pines actualmente solo son empleados 4, se espera que en un futuro el resto de pines se habiliten para otros sensores



1GND
2 sensor vel
3 pinza
8 placa rele

10. Otros ensayos realizados

10.1 Ensayo de velocidad del rotor según la frecuencia de alimentación.

HZ	RPM medida con una tacómetro	RPM medida con el microcontrolador
0	0	0
17.9	85	90
29	255	250
33.8	372	390
43.4	500	490
51.5	700	650
55.6	792	750
61.8	943	950
74.3	1180	1250
80.3	1277	1350
83.1	1362	1430
87.2	1476	1525
96.2	1743	1825
102.6	1870	1930
114.2	2125	2225
120	2221	2350

Tabla 2. Velocidades

10.2 ENSAYO DE VIBRACIONES

El maestro de taller que está realizando la campana de vacio me pidió que observara si el prototipo vibraba mucho cuando se encuentra en funcionamiento. Ya que se tiene la intención en el futuro de sustituir las patas actuales por unas ruedas para facilitar el trasporte y debajo de la placa en las que se sujetan las patas añadir otra placa de plástico sobre la que encajará herméticamente la campana. Con este ensayo quería ver si retirando las 4 patas las cuales cada una pesa 2.5 kilos el prototipo continuaba siendo estable, la función del peso de las patas era dar estabilidad al prototipo, si el prototipo no era estable a causa de no tener el peso que le confieren la patas habría que añadir alguna otra placa más pesada en la base para que cumpliera dicha función. Se llevo al motor hasta los 50 Hz, observándose que no existía ningún tipo de vibración.

Posteriormente a este ensayo se ha aumentado el rango de trabajo de 50 Hz a 130 Hz para darle mayor estabilidad al rotor en su giro y sea observado cierta vibración por lo que habría que pensar que si se sustituyen las patas habría que añadir peso para no comprometer la estabilidad del prototipo.

Lo que sí que se realizo fue cortar las espigas que sobresalían por debajo del prototipo para que las nuevas piezas que se están fabricando encajaran correctamente.

Ilustración 51. *Tornillos que sobresalen por la parte inferior*

Ilustraciones 52 y 53. *Detalle de los tornillos cortados*

11. CONCLUSIONES

Con la realización de este proyecto se ha conseguido que el prototipo funcione de forma automática sin tener que intervenir ningún operario. El microcontrolador gestiona todos los procesos a partir de la velocidad del prototipo o si el operario le da alguna orden de paro. Aunque no funciona tan bien como se esperaba, la levitación del rotor es muy escasa y llega a rozar en la parte inferior, este rozamiento hace que se frene más rápido de lo que en un principio se creía. Esta levitación tan escasa se debe a la instalacion del rodamiento el cual aumenta el peso del rotor. Además es posible que los superconductores por el paso del tiempo hayan perdido ciertas capacidades. Un futuro proyecto tendrá que estudiar la posibilidad de reducir el peso del rotor para aumentar la levitación.

Hasta ahora todas operaciones había que realizarlas manualmente, conectar y desconectar la alimentación del prototipo para mantenerlo en el rango de trabajo se realizaba manualmente con un interruptor trifásico. Ahora ya no es necesario que un operario retire la lamina de plástico ni los centradores sus funciones han sido adquiridas por la pinza. Además la pinza nos permite hacer pruebas sobre el prototipo sin necesidad de utilizar nitrógeno por lo que facilita los ensayos. Todas estas operaciones las gestiona el microcontrolador basándose en la velocidad que detecta gracias al sensor de velocidad. En la pantalla del microcontrolador ahora se puede ver la velocidad del rotor, no es la exacta ya que alguna vuelta se pierde mientras las contamos, pero su error es mínimo.

Se ha sustituido el variador de frecuencia de prácticas que se utilizaba desde el proyecto de Pedro J. Lambea por uno propio que será utilizado solo para el prototipo.

Además mediante este proyecto se ha dejado preparado para que en un futuro realizar modificaciones que completarían el prototipo tales como son la campana de vacío, las ruedas para el transporte. La intención de poner la campana de vacío se debe a eliminar los problemas actuales de creación de hielo en las partes frías del prototipo que después debemos secar con un secador para que no se oxiden y se deteriore además de que estos cristales provocan cierto rozamiento con el eje del rotor. Y de esta forma mejoraríamos el rendimiento del prototipo ya que no consumiría tanto nitrógeno. Y la posibilidad de realizar en el futuro modificaciones en el comportamiento del proyecto actualizando el software que ejecuta el microcontrolador, como pueden ser el programa del sensor de nivel de nitrógeno, de un sensor de temperatura o del programa que active la bomba de vacío.

Otro de los aspectos que se tendrá que tener en cuenta en proyectos siguientes es el diseñar y construir un nuevo rotor para aligerarlo de peso ya que las pastillas superconductoras a lo largo del tiempo están perdiendo sus propiedades y a causa de añadir peso al ser añadido el rodamiento. También se tiene la intención de modificar el rotor para que pueda actuar como generador

Personalmente decir que a pesar de los quebraderos de cabeza que he tenido para ir solucionando los problemas con los que me he encontrado a lo largo de la ejecución del proyecto, me siento satisfecho y contento de haberme embarcado en su realización. Ya que me ha permitido profundizar mis conocimientos en un gran abanico de campos relacionados con lo estudiado durante la carrera. Además de permitirme el manejo de

distintas herramientas que había en los talleres que al principio me daban reparo utilizarlas por miedo a estropearlas, gracias al proyecto en la actualidad me puedo manejar con ellas fluidamente y de forma segura tanto para mí como para las propias maquinas.

12. Bibliografía

(1) Juan Carlos Vesga Ferreira: *Micocontroladores Motorota-Freescale.* Marcombo. Barcelona, 2008.

(2) Luis Joyanes Aguilar e Ignacio Zahonero Martinez: Programación en C. Mc Graw Hill, Madrid, 2005

(3) *Apuntes de microcontroladores e instrumentación electrónica.* Universidad de Zaragoza. EUITIZ.

(4) *Apuntes de Regulación de maquinas electricas*. Universidad de Zaragoza. EUITIZ.

(5) *C++ Como si estuviera en primero*. Universidad de navarra

(6) Freescale Semiconductors: http://www.freescale.com

(7) RS Amidata: http://es.rs-online.com

(8) Farnell: http://es.farnell.com

(9) Todopic: http://todopic.mforos.com

(10) Wikipedia:http://www.wikipedia.org

(11) Curso de microcontroladores freescale: http://www.thinkchip.com.mx/index.php?option=com_content&view=article&id=46:demoac&catid=40:hcs08&Itemid=64

(12) µcontrol: http://www.ucontrol.com.ar/forosmf/freescale/sobre-tm0dh-tm0dl-valores-hexadecimales/

(13) Foros de electrónica: http://www.forosdeelectronica.com/

(14) R-Luis:http://r-luis.xbot.es/pic1/pic01.html

(14) Proyectos electronicos: http://proyectoselectronics.blogspot.com/2008/09/optoacoplador-que-es-y-como-funcionan.html

(15) Programar en C: http://elvex.ugr.es/decsai/c/

(16) Marco A. Peña Basurto & José M. Cela Espín: *Introducion a la programación en "C"*

(17) Lambea, P.J.: Prototipo de volante de inercia levitado mediante superconductores. Proyecto final de carrera de la Euitiz, 2007.

(18) Fco Javier Garcés Ferrer: Diseño y construcción del prototipo de un volante de inercia basado en la levitacion. Proyecto final de carrera de la Euitiz, 2007

(19) Óliver González Comeras: Regulación de velocidad de un prototipo de volante de inercia levitado mediante superconductores. Proyecto final de carrera de la Euitiz , 2009.

13. Anexos

13.1 Anexo imágenes realizadas en 3D:

Imágenes de las piezas en 3D realizadas mediante el Autocad 2009.

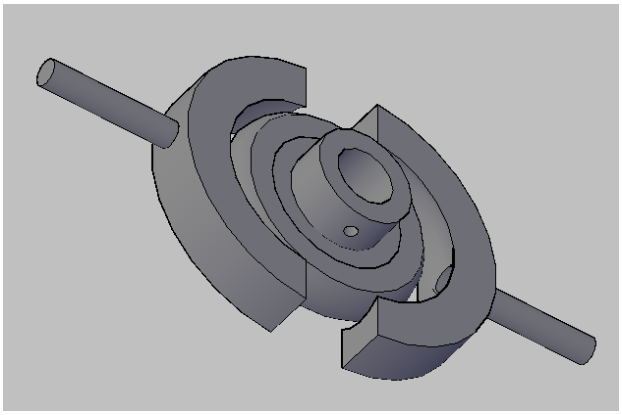


Imagen de las garras y el rodamiento, cuando la pinza se encuentra en estado abierto.

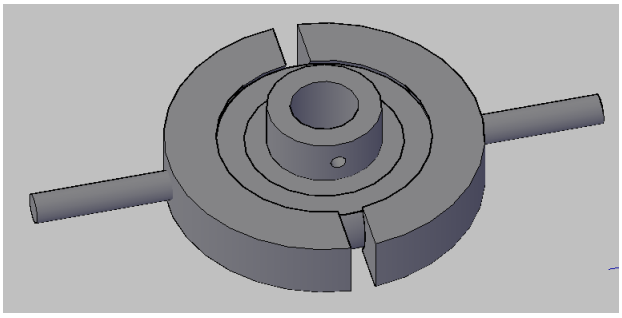
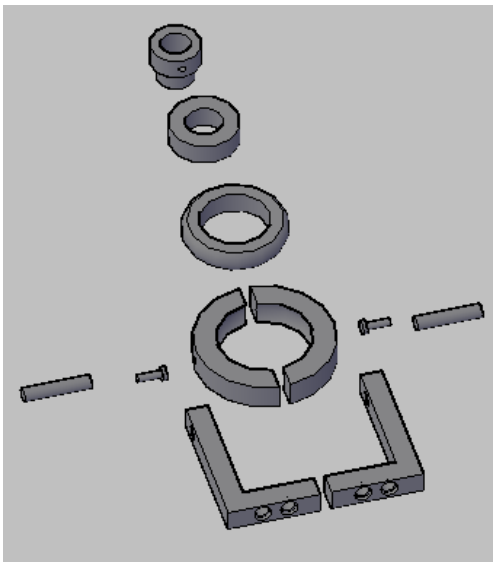
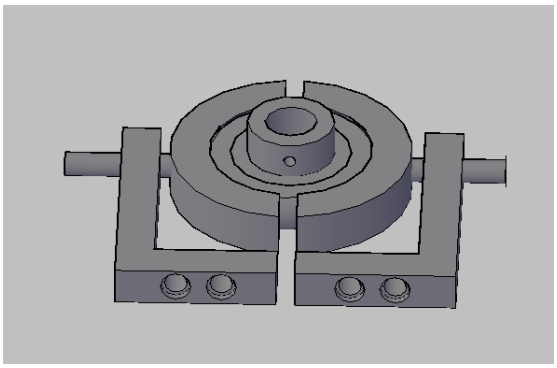


Imagen de las garras cuando se cierran sobre el rodamiento.

Imagen que muestra por orden de arriba hacia abajo: el casquillo, el rodamiento, la banda, las garras, los tornillos los fijadores y los dedos.

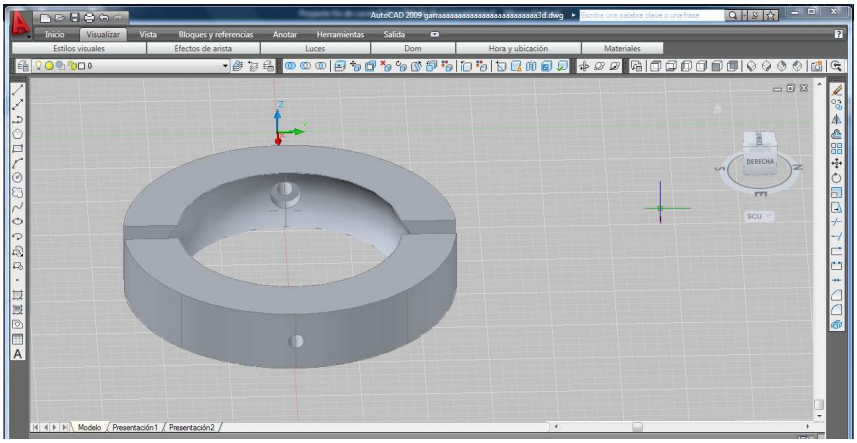


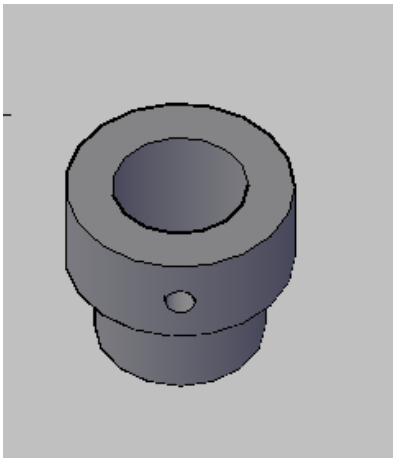


Pieza montada

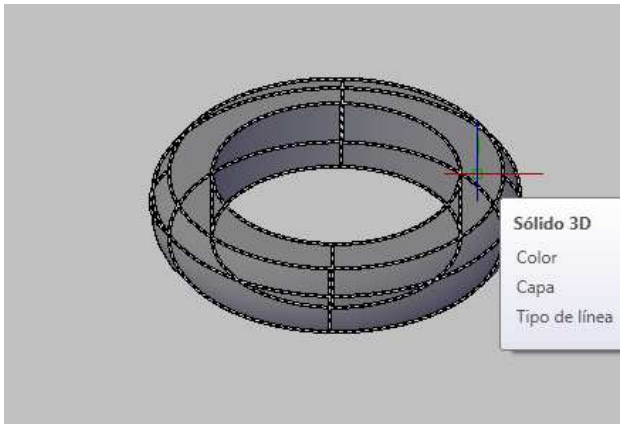
A continuación pueden verse en 3d las distintas piezas con mayor detalle.

La siguiente imagen corresponde a las garras. Se puede ver con detalle la forma cóncava con la que ha sido diseñada para que encaje perfectamente con la Banda y el agujero donde se alojará el tornillo que las une a los fijadores



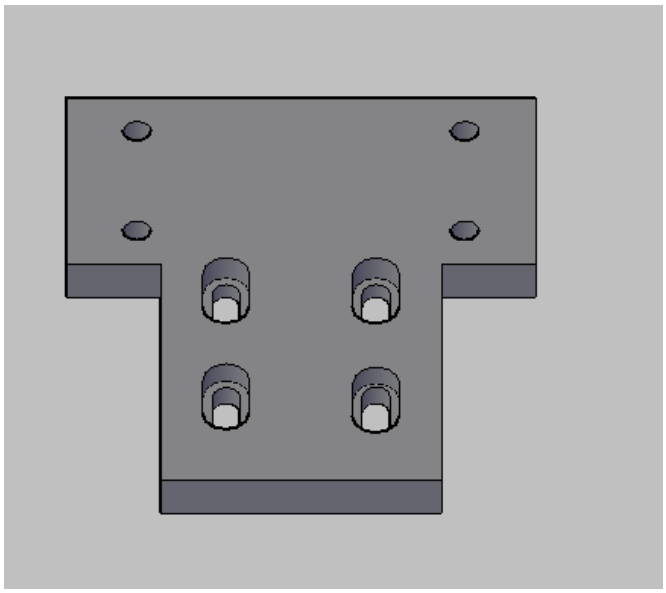


CASQUILLO



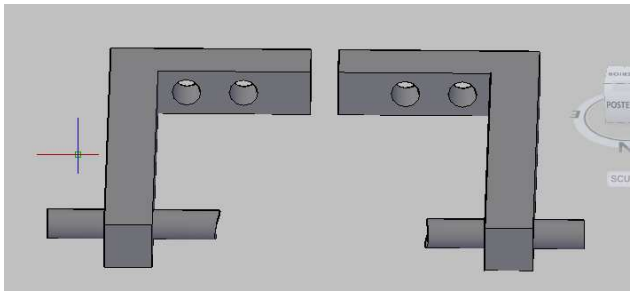
Banda del rodamiento

Pieza que se atornilla a la base superior del prototipo, esta pieza sirve para sujetar a la pinza que sostiene el rotor.

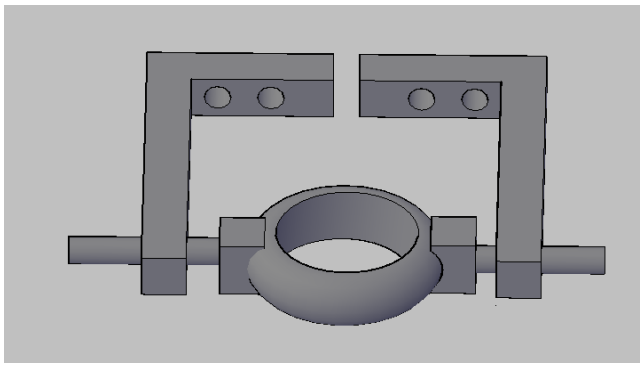


Versiones descartadas realizadas de las garras

Versión 1: descartada por no sujetar fuertemente el rotor cuando se alimenta el prototipo y por problemas para recuperar el rotor una vez abiertas.

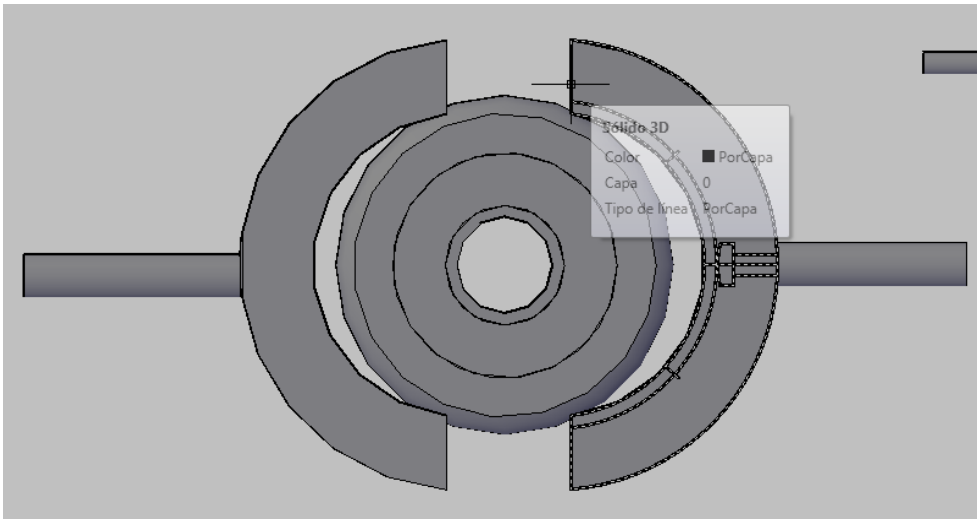


Versión 3: no se llego a fabricar ya que se opto por la versión 2 que es la que actualmente se usa en el proyecto. Posiblemente no funcionara ya que el campo magnético que atrae al rotor es más fuerte de lo que se esperaba por lo que tendría problemas similares a los de la versión 1.

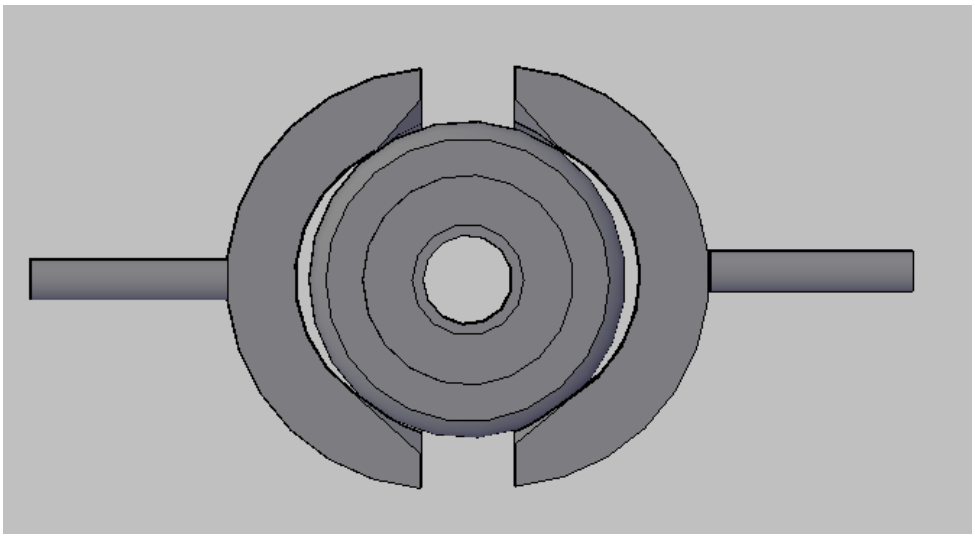


Modificación de la versión 2

Al abrirse la pinza, el rotor por su peso tiende a caer un poco hasta que encuentra el punto de equilibrio por los superconductores que provocan la levitación. Con el diseño de la versión 2 ocurría que el rotor quedaba suspendido por las cuatro esquinas de la garra, por lo que teníamos rozamiento lo cual es lo que se quiere eliminar. En la siguiente imagen puede observarse esta situación.



Para solucionar esto de avellanaron las esquinas.



13.2 Anexo líneas de código no empleados:

A continuación de escriben las líneas de código cuya finalidad era cambiar los datos de la variable velocidad de binario a decimal para ser mostrado en pantalla. No se consiguió que esta parte del código funcionara correctamente.

Código de la rutina binastrig

```
Void binastring( unsigned int value, char *str)
{
Char aux;
Unsigned int unit;
Aux=0;
Str[0] = str[1] = str[2] = str [3] = str[4] = “0”;
Str[5] = 0;
Unit  = 10000;
While (value)
{
If (value>=unit)
{
Value -=unit;
Str[aux]++;
}else
{
Aux++;
Unit /= 10;
}
```

Código de la rutina itoa

```
/* itoa: convierte datos , n, a cadenas de caracteres, s.*/
Void itoa (int n, char s [] )
{
Int i, sign;

If ((sign = n) < 0) /* graba sign*/
 n = -n:
i=0;
do (    /* genera digitos en orden inverso*/
s[i++] = n % 10 + `0`; /* obtener el digito siguiente*/
}while ((n /= 10) > 0);
If (sign < 0)
s[i++] = ´-´;
s[i] = ´/0´;
reverse (s);
}
```